UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE INGENIERIA

ELECTRÓNICA Y TELECOMUNICACIONES

TESIS

"SISTEMA DE RECONOCIMIENTO AUTOMATICO DE NUMEROS UTILIZANDO SIMILARIDAD DEL COSENO PARA VISIÓN ARTIFICIAL"

PRESENTADA POR:

SERRATO TALLEDO, JEANCARLO ALDEIR

PARA OPTAR EL TITULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO Y TELECOMUNICACIONES

Línea de investigación:

Sistemas Digitales

Piura, Perú

2017

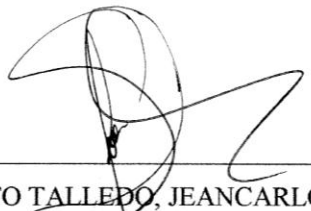
UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE CIENCIAS
ESCUELA PROFESIONAL DE INGENIERIA
ELECTRÓNICA Y TELECOMUNICACIONES

"SISTEMA DE RECONOCIMIENTO AUTOMATICO DE NUMEROS UTILIZANDO SIMILARIDAD DEL COSENO PARA VISIÓN ARTIFICIAL"

Línea de investigación:

Sistemas Digitales



Bch. SERRATO TALLEDO, JEANCARLO ALDEIR
EJECUTOR

Ing. JUAN M. JACINTO SANDOVAL
ASESOR

DECLARACIÓN JURADA DE AUTENTICIDAD DE LA TESIS

Yo: JEANCARLO ALDEIR SERRATO TALLEDO, identificado con CU: 1332010025 Y DNI N°73031223, Bachiller de Escuela Profesional de Ingeniería Electrónica y Telecomunicaciones, de la Facultad de Ciencias y domiciliado en Mz A lote 11 Los Medanos del Distrito de Castilla, Provincia de Piura, Departamento de Piura,

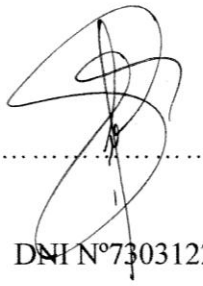
Celular 960391478

Email:jeancarlo137@gmail.com

DECLARO BAJO JURAMENTO: que la tesis que presento es auténtica e inédita, no siendo copia parcial ni total de una tesis desarrollada, y/o realizada en el Perú o en el Extranjero, en caso contrario de resultar falsa la información que proporciono, me sujeto a los alcances de lo establecido en el Art. N° 411, del código Penal concordante con el Art. 32° de la Ley N° 27444, y Ley del Procedimiento Administrativo General y las Normas Legales de Protección a los Derechos de Autor.

En fe de lo cual firmo la presente.

Piura Octubre del 2017



.....

DNI N°73031223

Artículo 411.- El que, en un procedimiento administrativo, hace una falsa declaración en relación a hechos o circunstancias que le corresponde probar, violando la presunción de veracidad establecida por ley, será reprimido con pena privativa de libertad no menor de uno ni mayor de cuatro años.

Art. 4. Inciso 4.12 del Reglamento del Registro Nacional de Trabajos de Investigación para optar grados académicos y títulos profesionales –RENATI Resolución de Consejo Directivo N° 033-2016-SUNEDU/CD

UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE INGENIERIA

ELECTRÓNICA Y TELECOMUNICACIONES

"SISTEMA DE RECONOCIMIENTO AUTOMATICO DE NUMEROS UTILIZANDO SIMILARIDAD DEL COSENO PARA VISIÓN ARTIFICIAL"

Línea de investigación:

Sistemas Digitales

APROBADA POR:

JURADO:

ING. FRANKLIN BARRA ZAPATA
PRESIDENTE

ING. CARLOS E. ARELLANO RAMIREZ
SECRETARIO

ING. MARIO A. RAMOS ECHEVARRIA
VOCAL

UNIVERSIDAD NACIONAL DE PIURA
FACULTAD DE CIENCIAS

ACTA DE SUSTENTACIÓN 070-2017-FC-UNP

Los Miembros del Jurado Calificador que suscriben, reunidos para evaluar la Tesis denominada **"SISTEMA DE RECONOCIMIENTO AUTOMÁTICO DE NÚMEROS UTILIZANDO SIMILARIDAD DEL COSENO PARA VISIÓN ARTIFICIAL"** presentado por el señor Bachiller **SERRATO TALLEDO – JEANCARLO ALDEIR,** con el asesoramiento del **Ing. Juan Manuel Jacinto Sandoval, M.Sc.**; oídas las observaciones y respuestas a las preguntas formuladas, y de conformidad al Reglamento de Tesis para obtener el Título Profesional en la Facultad de Ciencias, lo declaran:

APROBADO (X) **DESAPROBADO** ( )

Con la mención de:

MUY BUENO

(X) En consecuencia, queda en condición de ser ratificado por el Consejo de Facultad de Ciencias de la Universidad Nacional de Piura, y recibir el **TITULO PROFESIONAL DE INGENIERO ELECTRÓNICO Y TELECOMUNICACIONES.**

(X) En consecuencia, queda en condición de ser ratificado por el Consejo Universitario de la Universidad Nacional de Piura, y recibir el **TITULO PROFESIONAL DE INGENIERO ELECTRÓNICO Y TELECOMUNICACIONES**; después que el sustentante incorpore la sugerencia del Jurado Calificador.

Piura, 05 de octubre de 2017.

M.Sc. FRANKLIN BARRA ZAPATA
PRESIDENTE DE JURADO DE TESIS

Dr. CARLOS ENRIQUE ARELLANO RAMÍREZ
SECRETARIO DE JURADO DE TESIS

Ing. MARIO AUGUSTO RAMOS RAMÍREZ
VOCAL DE JURADO DE TESIS

UNIVERSIDAD NACIONAL DE PIURA DECANATO Facultad de Ciencias

DEDICATORIA

A mis padres, porque creyeron en mí y me sacaron adelante, dándome ejemplos dignos de superación y entrega

A mi abuelo Wilfredo y abuelas Rosa y María, por su apoyo incondicional

A mis hermanas Lucero, Estefany María, porque pesar de todo siempre están ahí cuando las necesito.

AGRADECIMIENTO

A mis padres, que me permitieron culminar mis estudios en esta carrera, asimismo me ayudaron, apoyaron y orientaron en cada momento de mi vida. Ambos son las personas más importantes en mi vida, y son la fuente de inspiración para mi trabajo, siempre estaré agradecido con ellos.

A mis hermanas, tíos, abuelos y amigos. Gracias por haber fomentado en mí el deseo de superación y anhelo en la vida.

Al Ing. Juan Manuel Jacinto Sandoval, por ser mi asesor en esta tesis, orientándome y ayudándome con sus conocimientos que fueron de vital importancia para el desarrollo de la presente tesis.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

El Reconocimiento de Patrones es el estudio de cómo las máquinas pueden observar el ambiente o entorno, aprender a distinguir patrones de interés a partir de la experiencia, y tomar decisiones razonables con respecto a las categorías a las que pertenecen dichos patrones. El mejor reconocedor de patrones conocido hasta ahora es el ser humano, no sabiéndose a ciencia cierta cuál es el proceso mediante el cual los humanos realizamos esta tarea. El Reconocimiento Óptico de Caracteres (OCR) es uno de los tópicos más antiguos dentro del Reconocimiento de Patrones y una de las áreas de investigación más importante y activa, que en la actualidad presenta desafíos: la precisión en el reconocimiento asociada tanto a caracteres impresos en una imagen degradada o a caracteres manuscritos es aún insuficiente, existiendo errores en el reconocimiento.

La etapa del reconocimiento se lleva a cabo, comparando las imágenes almacenadas y las imágenes capturadas. Para esto se leen previamente las imágenes almacenadas en formato de archivo. Se tienen modelos diferentes de cada número. Cuando el programa implementado lee los pixeles de cada número, los agrupa y promedia en pixeles, a este proceso se le denomina zonning, que lleva a reducir los datos. En la captura de la imagen que se desea procesar, se obtienen valores, pero previo procesamiento. Luego se comparan. A una de las plantillas se debe corresponder y quien hace esta comparación es la fórmula de Similaridad del coseno. Si el resultado que se obtiene después de aplicar la fórmula es cercano a 0, quiere decir que las imágenes de los números comparados no son iguales y si el resultado obtenido es cercano a 1 quiere decir que se trata de dos imágenes del mismo número.

Similaridad Coseno, números, imágenes, reconocimiento.

ABSTRACT

Pattern Recognition is the study of how machines can observe the environment or environment, learn to distinguish patterns of interest from experience, and make reasonable decisions regarding the categories to which those patterns belong. The best recognizer of patterns known so far is the human being, not knowing for sure what the process is by which humans perform this task. Optical Character Recognition (OCR) is one of the oldest topics within Pattern Recognition and one of the most important and active areas of research, which today presents challenges: the accuracy in recognition associated with both printed characters in A degraded image or handwritten characters is still insufficient, with errors in recognition.

The recognition step is performed by comparing the stored images and the captured images. For this the images stored in file format are read previously. They have different models of each number. When the implemented program reads the pixels of each number, groups them and averages them in pixels, this process is called zonning, which leads to reduce the data. In the capture of the image to be processed, values are obtained, but after processing. Then they compare. One of the templates must correspond and the one who makes this comparison is the similarity formula of the cosine. If the result obtained after applying the formula is close to 0, it means that the images of the numbers compared are not equal and if the result obtained is close to 1, it means that they are two images of the same number.

Similarity of cosine, number, images, recognition.

INTRODUCCIÓN

Es preciso reconocer que hoy por hoy la visión por computador a veces no es la mejor solución a un problema. Existen muchas ocasiones en las que el problema es tan complejo que la solución humana es lo mejor. Por ejemplo, imaginemos una conducción de un vehículo en una carretera con tráfico intenso. Pero a veces, las soluciones humanas tienden a ser inexactas o subjetivas y en ocasiones lentas y presentan una ausencia de rigor, así como una pobre percepción (Marshall y Martin 1993). No obstante, la solución humana es menos estructurada que la solución artificial y muchos problemas de visión por computador requieren un nivel de inteligencia mucho mayor que el que la máquina pueda ofrecer. El sistema de visión humana puede describir automáticamente una textura en detalle, un borde, un color, una representación bidimensional de una tridimensional, ya que puede diferenciar entre imágenes de diferentes personas, firmas, colores, etc., puede vigilar ciertas zonas, diagnosticar enfermedades a partir de radiografías, etc. Sin embargo, aunque algunas de estas tareas pueden llevarse a cabo mediante visión artificial, el software o el hardware necesario no consigue los resultados que serían deseables.

Aun a pesar de las limitaciones expuestas, cada día es mayor el número de aplicaciones de la visión artificial.

El reconocimiento de patrones, también llamado lectura de patrones, identificación de figuras y reconocimiento de formas, consiste en el reconocimiento de patrones de señales. Los patrones se obtienen a partir de los procesos de segmentación, extracción de características y descripción donde cada objeto queda representado por una colección de descriptores. El sistema de reconocimiento debe asignar a cada objeto su categoría o clase (conjunto de entidades que comparten alguna característica que la diferencia del resto). Para poder reconocer los patrones se siguen los siguientes procesos: adquisición de datos, extracción de características y toma de decisiones.

El punto esencial del reconocimiento de patrones es la clasificación: se quiere clasificar una señal dependiendo de sus características. Señales, características y clases pueden ser de cualquiera forma, por ejemplo, se puede clasificar imágenes digitales de letras en las clases «A» a «Z» dependiendo de sus píxeles o se puede clasificar ruidos de cantos de los pájaros en clases de órdenes aviares dependiendo de las frecuencias.

I. ASPECTOS DE LA PROBLEMATICA

1.1. DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA.

Los sistemas de reconocimiento automático existentes se basan en sistemas comerciales, y cada sistema tiene una metodología propia. Para hacer proyectos de investigación relacionados a visión artificial, donde involucren reconocimiento de caracteres, primero se necesita hacer un motor de búsqueda y comparación de caracteres, y como existen diferentes métodos, haremos uno propio.

¿Será posible implementar un sistema de visión artificial para reconocer números utilizando similaridad del coseno?

1.2. JUSTIFICACIÓN E IMPORTANCIA DE LA INVESTIGACIÓN

Los sistemas de visión artificial se utilizan principalmente en muchas aplicaciones de diferentes áreas, como en la identificación de placas de vehículos, etc. Este proyecto contribuiría con diferentes proyectos donde involucren sistemas OCR, siendo éste una etapa de un sistema completo, con algún propósito.

Existen muchos métodos de reconocimiento óptico de caracteres, como la distancia euclidiana, distancia Manhattan, distancia Mahalanobis, etc. Todas miden o comparan diferentes patrones, pero he optado por la similaridad del coseno, para ver qué tan eficiente y eficaz son los resultados que aporta.

Cabe resaltar, que la implementación de este sistema de visión artificial, junto con un software de procesamiento de los datos, agilizaría el tratamiento de la información, en muchas áreas, donde se procesen datos numéricos.

Es por estos motivos que se justifica la elaboración de este proyecto de tesis.

1.3. OBJETIVOS

OBJETIVO GENERAL

Diseñar un sistema de reconocimiento automático de números, utilizando similaridad del coseno para visión artificial.

OBJETIVOS ESPECÍFICOS

- Implementar técnicas de procesamiento para mejorar la imagen.
- Implementar algoritmos de extracción de características.
- Implementar el sistema utilizando archivos de imágenes.

1.4. DELIMITACIÓN DE LA INVESTIGACIÓN

El proyecto será desarrollado, en condiciones ambientales de iluminación y teniendo en cuenta pequeñas rotaciones de las imágenes de los números, además se harán capturas de imágenes teniendo en cuenta una sola distancia de la cámara con respecto a las imágenes. El proyecto se desarrollará en una computadora personal de características básicas.

II. MARCO TEORICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

En la Tesis “Reconocimiento de patrones utilizando técnicas estadísticas y conexionistas aplicadas a la clasificación de dígitos manuscritos” presentada por Leticia María Seijas, en la Universidad de Buenos Aires, Argentina, propone una nueva estrategia Bayesiana de combinación de clasificadores que permite detectar ambigüedades y resolverlas, lo que constituye la novedad y principal contribución de la tesis. Se propone, a su vez, un sistema completo de reconocimiento de patrones en dos niveles, con una arquitectura modular y paralelizable, que utiliza distintas características extraídas de los patrones de entrada según el problema a resolver junto con la estrategia Bayesiana ya mencionada que decide la respuesta del sistema. Como elementos componentes del reconocedor, en una primera capa o nivel, se utilizan clasificadores relativamente sencillos y bien posicionados para el problema a tratar. Los elementos pertenecientes a la segunda capa se utilizan para estimar cuán confiable es la respuesta de cada clasificador individual frente a un patrón de entrada, permitiendo decidir cuándo un patrón debe ser considerado bien definido o ambiguo, y en este último caso con qué clases podría confundirse. Adicionalmente, se proponen y aplican estrategias de selección de clasificadores en la etapa de construcción del reconocedor.

El sistema reconocedor de patrones presentado fue aplicado al problema del reconocimiento de dígitos manuscritos off-line, como forma de testear su desempeño. En función de esto, se proponen descriptores basados en características de multirresolución a través del uso de la Transformada Wavelet CDF y de Análisis de Componentes Principales, que permiten disminuir considerablemente el tamaño del patrón de entrada y aumentar la calidad de la representación.

La experimentación se realizó sobre las bases de datos CENPARMI y MNIST, ampliamente referenciadas para este problema. Se obtuvieron altos porcentajes en el reconocimiento que alcanzaron un 97,40 y 99,32% para las bases CENPARMI y MNIST respectivamente. Dichos valores son comparables a los resultados publicados considerados representativos.

Los sistemas existentes, este tipo de herramientas normalmente son una variación de las herramientas de reconocimiento de patrones, que han sido específicamente mejorados para el reconocimiento de caracteres. Las técnicas utilizadas pueden ser la de búsqueda de patrones o la de detección de contornos. Una de las características a tener en cuenta, en este tipo de software, es la facilidad con que se realiza el aprendizaje de las nuevas fuentes de letras.

En este caso, los algoritmos basados en los contornos son ideales para imágenes donde los caracteres estén muy bien contrastados con respecto al fondo, y de esta forma se pueden leer caracteres girados o ampliados con respecto a los caracteres aprendidos.

Las herramientas de reconocimiento de caracteres basados en reconocimiento de patrones por correlación de niveles de gris proporcionan una detección más robusta en aplicaciones donde el contraste y la iluminación puedan variar, sin embargo, son menos útiles si los caracteres están rotados o el tamaño vario. En estos casos el entrenamiento previo de los caracteres debe ser más intensivo.

2.2. BASES TEÓRICAS

2.2.1. RECONOCIMIENTO ÓPTICO DE CARACTERES[1]

En los últimos años la digitalización de la información ya sea: textos, imágenes, sonido, etc. Se ha vuelto un gran punto de interés para la sociedad, es por eso que, en el caso de los textos, se generan continuamente grandes cantidades de información escrita, tipográfica o manuscrita en todo tipo de soporte. En este contexto, dentro del reconocimiento de caracteres el poder automatizar la introducción de caracteres evitando la entrada por teclado, implica un importante ahorro de recursos humanos y un aumento de la productividad, al mismo tiempo que se mantiene, o hasta se mejora, la calidad de muchos servicios.

Por consiguiente, el reconocimiento óptico de caracteres genera como principal objetivo la tarea de poder identificar automáticamente símbolos

[1] https://es.wikipedia.org/wiki/Reconocimiento_%C3%B3ptico_de_caracteres

o caracteres que pertenecen a un determinado alfabeto, a partir de una imagen para almacenarla en forma de datos con los que se podrá interactuar mediante un programa de edición de texto o similar.

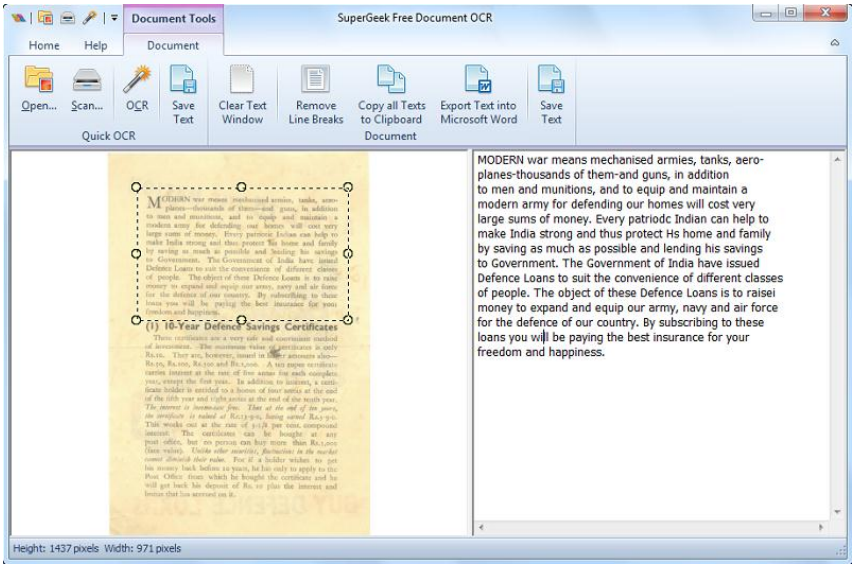


FIGURA 1: SISTEMA DE RECONOCIMIENTO ÓPTICO DE CARACTERES

2.2.1.1. QUE SIGNIFICA OCR

Reconocimiento Óptico de Caracteres, con siglas ROC, o llamado en inglés OCR, es una tecnología que le permite que diferentes tipos de documentos, tales como papeles escaneados, archivos PDF o imágenes captadas por una cámara digital se conviertan en datos con opción de búsqueda y funcionalidad de editar.

El software OCR verifica la imagen pixel por pixel buscando formas que coincidan con los caracteres buscados, en función de la complejidad este buscara diferentes coincidencias con los mismos, mediante un determinado número de fuentes., el OCR puede analizar los elementos ya sean bloques de texto, imágenes o tablas, verificando los espacios en blanco, descomponiendo el texto en líneas, palabras y caracteres de forma que el programa pueda realizar el reconocimiento mediante los diccionarios que posee internamente, y dar con el valor exacto o más cercano buscado.

2.2.1.2. PROCESO[2]

El proceso de OCR se inicia a partir del análisis de esta imagen que dará como resultado la conversión de los conjuntos de puntos en caracteres manipulables como los archivos producidos en cualquier tratamiento de textos. Una vez acabada esta parte, se inicia una fase de chequeo de los resultados para terminar exportando los resultados a otro tipo de aplicación (tratamientos de texto, programas de edición, hojas de cálculo, entre otras aplicaciones).

Mientras se ejecutan las acciones del reconocimiento de caracteres, los programas analizan las imágenes de los párrafos escritos para producir el correspondiente texto editable. Después de acabar el proceso, se puede exportar el texto reconocido, mediante los pertinentes formatos, a toda la amplia variedad de aplicaciones, que trabajan con el producto resultante.

2.2.1.3. CARACTERÍSTICAS

Son caracteres impresos cuya información es leída automáticamente por un haz de luz y decodificados por algoritmos matemáticos a una forma digital, analógica o ASCII.

La lectura es por contacto a distancia, el haz es fijo o móvil y visible o no (infrarrojo). Estos sistemas están siendo desplazados por el Código de Barras, para su uso comercial masivo.

El código de barras no es un sistema automático de identificación ya que tanto la marcación como la lectura de cada producto es manual y el sistema OCR se refiere solamente a los sistemas que leen automáticamente la información.

2.2.1.4. OBJETIVO DEL OCR

Los sistemas de reconocimientos ópticos de caracteres, así como los de reconocimientos de texto en general, tienen como objetivo automatizar la introducción de caracteres al sistema, evitando la entrada

[2] https://www.ecured.cu/Reconocimiento_%C3%B3ptico_de_caracteres

por el teclado, implicando un importante ahorro de recursos para las empresas y un incremento en la productividad en el mismo tiempo que se preserva o mejora la calidad de los servicios ofrecidos al cliente.

2.2.1.5. VARIANTES DE SISTEMAS OCR

Existen sistemas OCR muy diversos según los tipos de problemas que abordan y las funcionalidades que ofrecen. Existe una terminología de empleo extendido en la industria para referirse a cada una de las variantes específicas de sistemas OCR.

- ICR (Intelligent Character Recognition): se utiliza para designar el reconocimiento de caracteres manuscritos.
- OCV (Optical Character Verification): hace referencia a la verificación de contenidos previamente conocidos.
- OMR (Optical Mark Recognition): designa una funcionalidad de reconocimientos de marca.

2.2.1.6. PROBLEMÁTICA DE LOS OCR

El problema del reconocimiento óptico de caracteres ha estado abordado de manera intensa por disciplinas científicas, como el "Reconocimiento de formas" y "Visión artificial". Las características de este problema y las diversas vertientes que presenta, hacen que sea un proceso en continua investigación. Desde el reconocimiento de caracteres impresos, de uno o múltiples tipos de letras (considerado en la práctica como un problema superado), como la escritura continua, restringida o no (de gran dificultad intrínseca), pasando por el reconocimiento de caracteres manuscritos aislados o la disposición fija o flotante del texto a reconocer, son unos de los problemas más frecuentes en los sistemas OCR.

2.2.1.7. Desventajas de sistemas anteriores

El sistema usado hasta no hace mucho consistía en que el OCR aislaba la imagen correspondiente a un carácter y la comparaba con una base de caracteres para determinar su correspondiente código ASCII o

bien Unicode. Una vez reconocido lo transformaba y lo trasladaba al documento OCR resultante. Este sistema tenía el inconveniente de que trabajaba con un número limitado de fuentes, por lo que el resultado obtenido no siempre era el deseado.

2.2.1.8. Ventajas de sistemas actuales

Los programas actuales de OCR están basados en el análisis de características de los caracteres en vez de en la coincidencia de las matrices de estos, lo que permite una mayor velocidad en el proceso y el no tener que depender de una limitada base de fuentes.

Hay en el mercado bastantes programas de OCR, entre los que cabe destacar los conocidos OmniPage, Abbyy Fine Reader o READiris. Versiones reducidas de estos programas suelen contarse entre el software incluido en los escáneres.

El OmniPage Professional por su amplia variedad de opciones y su buena presentación y funcionamiento.

Estos programas son de especial utilidad cuando necesitamos hacer referencia a textos en un escrito (copiar literalmente un texto dentro de otro), como puede ser el caso de redacción de informes o referencias bibliográficas.

También pueden ser utilizados en el mundo de la música, ya que la mayoría de ellos tienen también capacidad de leer partituras musicales.

Actualmente estos programas son capaces de reconocer no solo el texto en sí, sino también el estilo y formato de este, aunque dentro de unas limitaciones, haciendo necesario que posteriormente editemos el texto resultante y revisemos estos dos parámetros.

2.2.1.9. Requerimientos para un buen funcionamiento

Para su correcto funcionamiento es necesario que la imagen de donde provenga dicho texto esté en las mejores condiciones posibles para que pueda reconocer correctamente los caracteres. Hay que tener en cuenta que factores tales como un texto borroso (aunque sea solo ligeramente),

papel manchado o demasiado fino, arrugas o arañazos en el documento, falta de una parte de una letra, cualquier tipo de transparencia en el papel, entre otros aspectos, van a dificultar el correcto reconocimiento de este texto.

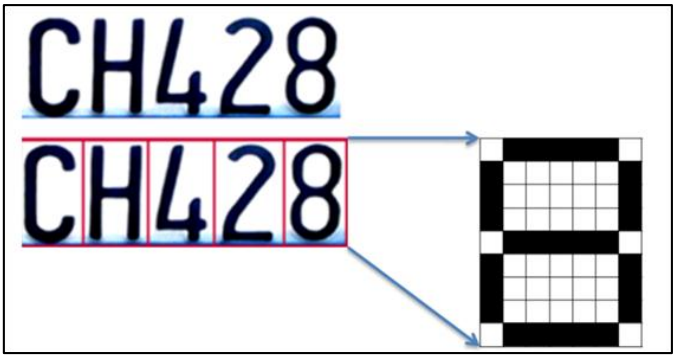
El promedio de efectividad de los programas OCR, en perfectas condiciones, ronda el 90%, disminuyendo de forma ostensible al disminuir la calidad del original. A esto hay que añadir una correcta configuración del escáner tanto en su resolución como en su brillo y en la limpieza de la lente y cristal. Uno de los requisitos básicos para que un programa OCR funcione correctamente es que necesita una imagen de gran calidad.

Es cierto que también pueden reconocer textos escritos manualmente, pero en este caso siempre a condición de que estén escritos claramente, a ser posible con letra de molde, y que esta se asemeje lo más posible a un tipo de letra existente.

2.2.2. ESQUEMA BÁSICO DE UN ALGORITMO DE ROC[3]

Todos los algoritmos de ROC tienen la finalidad de poder diferenciar un texto de una imagen cualquiera. Para hacerlo se basan en cuatro etapas:

- Binarización.
- Fragmentación o segmentación de la imagen.
- Adelgazamiento de los componentes.
- Comparación con patrones.



[3] https://es.wikipedia.org/wiki/Reconocimiento_%C3%B3ptico_de_caracteres

FIGURA 2: CARÁCTER REPREENTADO EN PIXELES

2.2.2.1. Binarización

La mayor parte de algoritmos de ROC parten como base de una imagen binaria (dos colores). Por lo tanto, es conveniente convertir una imagen de escala de grises, o una de color, en una imagen en blanco y negro, de tal forma que se preserven las propiedades esenciales de la imagen. Una forma de hacerlo es mediante el histograma de la imagen, donde se muestra el número de píxeles para cada nivel de grises que aparece a la imagen. Para binarizarla tenemos que escoger un umbral adecuado, a partir del cual todos los píxeles que no lo superen se convertirán en negro y el resto en blanco.

Mediante este proceso obtenemos una imagen en blanco y negro donde quedan claramente marcados los contornos de los caracteres y símbolos que contiene la imagen. A partir de aquí podemos aislar las partes de la imagen que contienen texto (más transiciones entre blanco y negro).

2.2.2.2. Fragmentación o segmentación de la imagen

Este es el proceso más costoso y necesario para el posterior reconocimiento de caracteres. La segmentación de una imagen implica la detección mediante procedimientos de "etiquetado determinista" o estocástico de los contornos o regiones de la imagen, basándose en la información de intensidad o información espacial.

Permite la descomposición de un texto en diferentes entidades lógicas, que han de ser suficientemente invariables, para ser independientes del escritor, y suficientemente significativas para su reconocimiento.

No existe un método genérico para llevar a cabo esta segmentación de la imagen que sea lo suficientemente eficaz para el análisis de un texto. Aunque las técnicas más utilizadas son variaciones de los métodos basados en proyecciones lineales.

Una de las técnicas más clásicas y simples para imágenes de niveles de grises consiste en la determinación de los modos o agrupamientos

(clústeres) a partir del histograma, de tal forma que permitan una clasificación o umbralización de los píxeles en regiones homogéneas.

2.2.2.3. Adelgazamiento de los componentes

Una vez aislados los componentes conexos de la imagen, se les tendrá que aplicar un proceso de adelgazamiento para cada uno de ellos. Este procedimiento consiste en ir borrando sucesivamente los puntos de los contornos de cada componente de forma que se conserve su tipología.

La eliminación de los puntos ha de seguir un esquema de barridos sucesivos para que la imagen continúe teniendo las mismas proporciones que la original y así conseguir que no quede deforme.

Se tiene que hacer un barrido en paralelo, es decir, señalar los píxeles borrables para eliminarlos todos a la vez. Este proceso se lleva a cabo para hacer posible la clasificación y reconocimiento, simplificando la forma de los componentes.

2.2.2.4. Comparación con patrones

En esta etapa se comparan los caracteres obtenidos anteriormente con unos teóricos (patrones) almacenados en una base de datos. El buen funcionamiento del ROC se basa en gran medida en una buena definición de esta etapa.

Existen diferentes métodos para llevar a cabo la comparación. Uno de ellos es el método de proyección, en el cual se obtienen proyecciones verticales y horizontales del carácter por reconocer, y se comparan con el alfabeto de caracteres posibles hasta encontrar la máxima coincidencia.
Existen otros métodos, como, por ejemplo:

- Métodos geométricos o estadísticos.
- Métodos estructurales.
- Métodos neuromiméticos.
- Métodos markovianos (modelo oculto de Márkov).
- Métodos de Zadeh.

2.2.2.5. Aplicaciones

Desde la aparición de los algoritmos de ROC, han sido muchos los servicios que han introducido estos procesos para aumentar su rendimiento y otros que se basan completamente en estas tecnologías. A continuación, se muestran algunas de las más destacables aplicaciones que utilizan el ROC.

Reconocimiento de texto manuscrito

Las dificultades que podemos encontrar a la hora de reconocer un texto tipografiado, no se pueden comparar con las que aparecen cuando queremos reconocer un texto manuscrito.

El reconocimiento de un texto manuscrito continúa siendo un desafío. Aunque el texto se compone básicamente de caracteres individuales, la mayoría de algoritmos ROC no consiguen buenos resultados, ya que la segmentación de texto continuo es un procedimiento complejo.

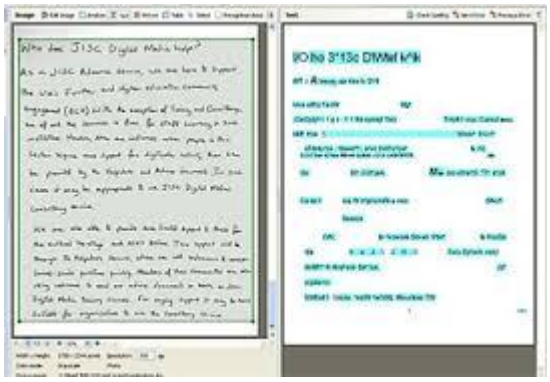


FIGURA 3: RECONOCIMIENTO DE TEXTO MANUSCRITO

En el caso de reconocimiento de escritura manuscrita a la hora de corrección de exámenes, existe la posibilidad, añadiendo un listado de léxico (nombres y apellidos) de acercarse al 100% de acierto. A través de las casillas de respuesta ICR se pueden reconocer palabras, como nombres de países, nombres de regiones, marcas comerciales, en resumen, todo aquello que pueda ser integrado en una lista de palabras (léxico), el cual puede ir aumentándose según las necesidades.

Por otro lado, se puede llegar a comprender una frase cuando la hemos terminado de leer. Esto implica una operación de niveles morfológico, léxico y sintáctico que se consigue mediante el reconocimiento del habla continua. Para llevar a cabo esa metodología, se utilizan algoritmos robustos que usan una segmentación previa, debido a que se obtiene automáticamente con la descodificación.

Reconocimiento de matrículas

Una de las aplicaciones son los radares. Estos deben ser capaces de localizar una matrícula de un vehículo con condiciones de iluminación, perspectiva y entorno variables.

En la etapa de segmentación, se buscan texturas similares a la de una matrícula y se aísla el área rectangular que forma la matrícula.

Finalmente, se aplica un proceso de clasificación múltiple sobre el conjunto de píxeles pertenecientes a la matrícula, proporcionando una cadena de caracteres que se tienen que ajustar a un modelo conocido: el formato de una matrícula. Si aparece algún error, es corregido.

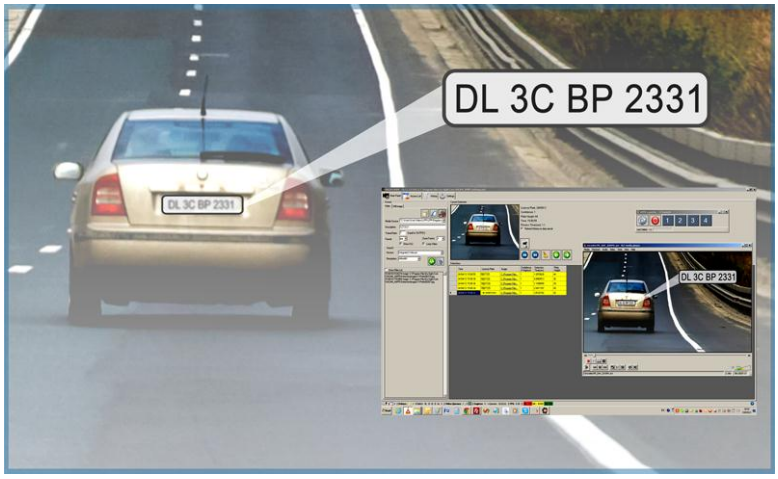


FIGURA 4: RECONOCIMIENTO DE MATRICULAS

Indexación con bases de datos

Con el gran aumento de información publicada que ha tenido lugar en los últimos años, cada vez son más los métodos que se emplean para organizar todo este material almacenado en bases de datos. Uno de

estos contenidos son las imágenes. Una de las formas más corrientes de buscar imágenes es a partir de metadatos introducidos manualmente por los usuarios. Actualmente han aparecido buscadores que proporcionan la posibilidad de buscar imágenes mediante el texto que aparecen en ellas, como el buscador DIRS (Document Image Retrieval System) que, mediante un algoritmo de ROC, extrae el texto que aparece en la imagen y lo utiliza como metadato que podrá servir para las búsquedas. Esta tecnología proporciona una posibilidad en la búsqueda de imágenes y demuestra que el ROC aún puede dar mucho de sí.

Reconocimiento de datos estructurados con ROC Zonal

Se usa para digitalizar de forma masiva grandes cantidades de documentos estructurados o semiestructurados (facturas, nóminas, albaranes, pólizas, justificantes bancarios, etcétera), catalogando automáticamente los documentos con los metadatos obtenidos y archivándolos en formato digital de forma indexada para facilitar su posterior búsqueda. Tiene el inconveniente de que es necesario diseñar previamente las plantillas, pero con una buena configuración se ahorra mucho tiempo en el proceso de digitalización.

2.2.3. SIMILITUD COSENO

La similitud coseno es una medida de la similitud existente entre dos vectores en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es cero, es decir si ambos vectores apuntan a un mismo lugar. Cualquier ángulo existente entre los vectores, el coseno arrojaría un valor inferior a uno. Si los vectores fuesen ortogonales el coseno se anularía, y si apuntasen en sentido contrario su valor sería -1. De esta forma, el valor de esta métrica se encuentra entre -1 y 1, es decir en el intervalo cerrado [-1,1].

Esta distancia se emplea frecuentemente en la búsqueda y recuperación de información representando las palabras (o documento) en un espacio vectorial. En minería de textos se aplica la similitud coseno con el objeto de establecer una

métrica de semejanza entre textos. En minería de datos se suele emplear como un indicador de cohesión de clusteres de textos. La similitud coseno no debe ser considerada como una métrica debido a que no cumple la desigualdad triangular.

En este caso la similitud de dos vectores de datos es el valor absoluto del coseno del ángulo que forman esos dos vectores (producto escalar entre vectores).

$$d_{coseno}(u,v) = \frac{|u \cdot v|}{\|u\|\|v\|} = \frac{\sum_{i=1}^{n} u_i v_i}{(\sqrt{\sum_{i=1}^{n} u_i{}^2})(\sqrt{\sum_{i=1}^{n} v_i{}^2})}$$

FIGURA 5 FORMULA SIMILITUD DEL COSENO

La virtud de utilizar el valor absoluto de esta distancia es que el método no depende de las ganancias de amplificación de los valores ni de los signos de polaridad de los mismos.

Dos vectores se pueden representar en el hiperplano y medir las diferencias de dirección entre ambos de la misma forma que se haría en un plano bidimensional, es decir, comparando el coseno del ángulo que forman.

Pongamos por ejemplo dos vectores bidimensionales:

$\vec{a} = (1,1)$

$\vec{b} = (3,0)$

Como podemos observar en siguiente figura existe una desviación importante entre la dirección que toma cada uno de estos vectores.

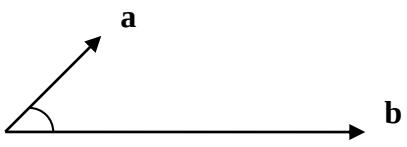


FIGURA 6 VECTORES CON DIRECCIONES DIFERENTES

Podemos medir esta desviación por el ángulo que forman los vectores, o por una medida proporcional al mismo, el coseno del ángulo que forman dichos vectores. Esta medida nos proporciona una estimación bastante acertada de la similitud o disimilitud de los vectores o lo que es lo mismo, de los documentos que representan.

Si representamos un vector con una longitud euclídea de uno sobre un eje de coordenadas, como en la figura siguiente:

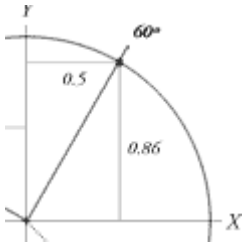


FIGURA 7 VECTOR SOBRE EJE X DE COORDENADAS

Tenemos que el coseno del ángulo que forma dicho vector con respecto del eje de las X se puede obtener hallando el cateto más próximo del triángulo rectángulo resultante. En la figura anterior se puede observar cómo el cateto más próximo mide 0,5.

En el caso de que la hipotenusa fuera 1, el coseno del ángulo sería equivalente a la longitud de dicho cateto. En el caso de que la hipotenusa fuera diferente de 1 sería necesario normalizar, y diríamos que el coseno = cateto más próximo / hipotenusa. El cateto más próximo normalizado por la hipotenusa, en otras palabras.

La ecuación necesaria para medir el coseno en un caso genérico (independientemente de la longitud y dimensiones de los vectores) se expresa como:

$$cos(\vec{a}, \vec{b}) = \frac{\vec{a} * \vec{b}}{|\vec{a}| * |\vec{b}|}$$

FIGURA 8 ECUACIÓN DE SIMILITUD DEL COSENO

Esto produce un valor entre 0 y 1, donde 0 indica la perpendicularidad (ortogonalidad) de los vectores, tendríamos dos vectores de dirección completamente distinta, y 1 indica que los vectores tienen idéntica dirección. En nuestro caso esta fórmula nos proporciona un resultado de 0,68.

2.2.4. VISION ARTIFICIAL

También conocida como visión por computador (del inglés computer vision) o visión técnica, es un subcampo de la inteligencia artificial y es el campo de acción más ambicioso del procesamiento digital de imágenes. Básicamente el objetivo es automatizar funciones de inspección visual, tradicionalmente utilizadas por el hombre como, por ejemplo.

Los objetivos típicos de la visión artificial incluyen:

La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas).

La evaluación de los resultados (ej.: segmentación, registro).

Registro de diferentes imágenes de una misma escena u objeto, hacer concordar un mismo objeto en diversas imágenes.

Seguimiento de un objeto en una secuencia de imágenes.

Mapeo de una escena para generar un modelo tridimensional de la escena; tal modelo podría ser usado por un robot para navegar por la escena.

Estimación de las posturas tridimensionales de humanos.

Búsqueda de imágenes digitales por su contenido.

2.2.5. PROCESAMIENTO DE IMAGENES[4]

Imagen: Es la proyección en perspectiva en el plano bidimensional de una escena tridimensional en un determinado instante de tiempo t_0.

[4] Rafael C. Gonzales, Richard E. Woods, "Tratamiento Digital de Imágenes"

Fotograma: Es una matriz bidimensional de valores de intensidad lumínica obtenidos para un tiempo t0 constante. Pudiera decirse en cierta forma que es una imagen discretizada.

Píxel (Picture Element): Es cada de una de las posiciones en que es discretizada una imagen, o lo que es lo mismo, cada una de las posiciones de un cuadro.

Imagen binaria: Son aquellas imágenes cuyos píxeles solo tienen dos valores: cero y uno.

2.2.6. LA IMAGEN DIGITAL[5]

Son el principal ingrediente de lo que se conoce como Visión Artificial y representan mediante algún tipo de codificación, normalmente en una matriz de números de dos dimensiones, una escena del entorno.

Existen dos tipos de imágenes utilizadas frecuentemente en Visión Artificial: imágenes de intensidad e imágenes de alcance (también llamadas imágenes de profundidad o perfiles de superficie). Las imágenes de intensidad miden la cantidad de luz que incide en un dispositivo fotosensible, mientras que las imágenes de alcance estiman directamente la estructura en tres dimensiones (3D) de la escena ya que su fundamento radica en el uso de sensores de alcance ópticos y algún fenómeno físico para adquirir la imagen. Un ejemplo típico de una imagen de intensidad es una fotografía, mientras que de una imagen de alcance es, por ejemplo, la imagen que obtiene el oftalmólogo sobre el grado de rugosidad de la córnea de un paciente o las imágenes de un radar.

Aunque la filosofía de los diferentes tipos de imágenes es diferente, en cualquier caso, tras su captura tendremos una matriz de valores en dos dimensiones (2D), es decir, una imagen digital.

2.2.7. DISPOSITIVOS DE CAPTURA DE IMÁGENES

Para la adquisición de imágenes digitales se requieren dos elementos básicos. El primero es un dispositivo físico que es sensible a una determinada

[5] Gonzalo Pajares Martinsanz, Jesús Manuel de la Cruz García, José Manuel molina pascual, Juan Cuadrado Pardo, Alejandro López Correa, “Imágenes digitales - Procesamiento practico con java”

banda del espectro de energía electromagnético (tal como rayos X, ultravioleta, visible, infrarrojo, etc.) y que produce una señal eléctrica de salida proporcional al nivel de energía incidente en cualquier instante de tiempo. El segundo, denominado digitalizador, es un dispositivo que cumple la función de convertir la señal eléctrica continua de salida del dispositivo físico en un conjunto discreto de localizaciones del plano de la imagen y, después, en la cuantización de dicha muestra. Esto implica, en primer lugar, determinar el valor de la imagen continua en cada una de las diferentes localizaciones discretas de la imagen (cada valor localizado de forma discreta se denomina muestra de la imagen) y, luego, asignar a cada muestra una etiqueta entera discreta, que es representativa del rango en el que varía la muestra.

Una vez capturada la señal continua y cuantificada espacialmente y en amplitud, se obtiene una imagen digital, que es como se representa en el computador, es decir, tendremos la matriz (2D) de números como ya hemos mencionado anteriormente. Éstos son los valores que se manipulan para extraer información de las imágenes mediante programas (software).

En la siguiente figura se ilustra un ejemplo de la cuantización espacial y en amplitud:

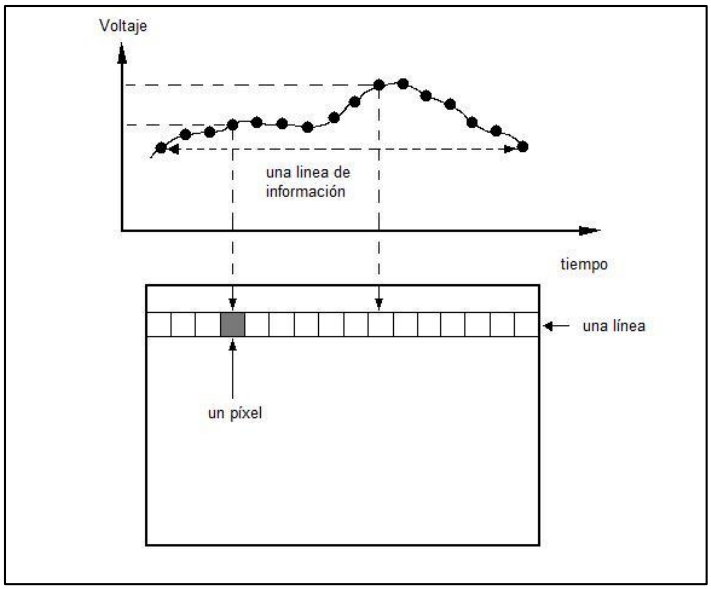


FIGURA 9: DIGITALIZACIÓN DE UNA SEÑAL ANALÓGICA

Supongamos que tenemos una señal analógica, que bien podría ser una línea de video analógica. Esta señal (línea) analógica de video se convierte a una imagen digital muestreando la señal analógica a intervalos determinados. El procedimiento consiste en medir el voltaje de la señal a intervalos de tiempo fijos. El valor del voltaje en cada instante se convierte a un número que es almacenado y se corresponde con la intensidad de la imagen en ese punto. La intensidad en cada punto depende tanto de las propiedades intrínsecas del objeto que se está viendo como de las condiciones de luz de la escena. Repitiendo este procedimiento para todas las líneas de video que constituyen una imagen, se pueden grabar los resultados obtenidos en el computador, de suerte que habremos conseguido una imagen digital que, en definitiva, es una matriz de números.

La imagen puede accederse como una matriz bidimensional (2D) de datos, donde cada punto o dato se denomina píxel.

Además de las cámaras de televisión que generan una imagen de video, uno de los sensores más usados para la visión artificial son loa dispositivos de acoplamiento de carga (Charge Coupled Devices - CCD). Entre los dispositivos CCD, que generalmente también producen una señal continua de video, cabe distinguir dos categorías: sensores de exploración de línea y sensores de exploración de área. Estos sensores CCD se basan en unos elementos semiconductores llamados photosites. Los fotones procedentes de la escena excitan el elemento semiconductor, de forma que el grado de excitación es proporcional a la cantidad de carga acumulada en el photosite y, por lo tanto, a la intensidad luminosa en ese punto.

Estos photosites se pueden representar en forma de matriz como se muestra en la siguiente figura:

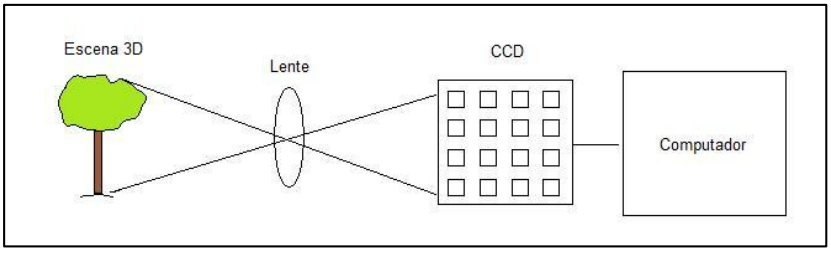


FIGURA 10: CAPTURA DE UNA IMAGEN 3D POR UN DISPOSITIVO CCD

Supongamos una matriz de photosites situados detrás de una lente y sobre los que se proyecta una imagen procedente de la escena 3D. La señal de estos sensores se procesa en el propio sensor (cámara) o en otro dispositivo (tarjeta de procesamiento de imágenes digitales) y los valores digitales se envían al computador.

Para clarificar un poco más estos conceptos, analicemos el ejemplo sencillo mostrado en la figura siguiente:

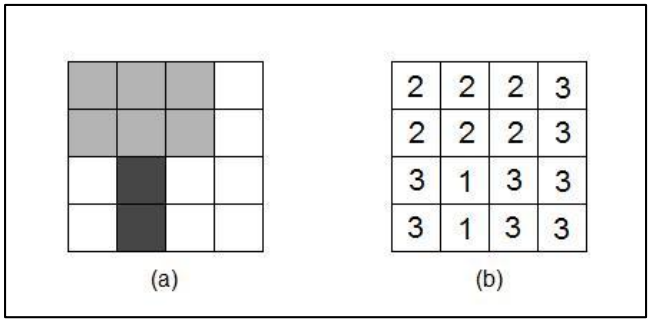


FIGURA 11: FIGURA DEL ÁRBOL CAPTURADA POR UNA CÁMARA CON 4X4 SENSORES DE INTENSIDAD

Supongamos una escena 3D cuya representación digital en forma de matriz, tal como se almacena en el computador, resulta ser la que se muestra en la figura 3-(b). En este caso se utiliza una matriz de 16 elementos (4x4) para representar el árbol; lógicamente, cuantos más sensores dispongamos en la cámara, más precisión podemos tener en la reproducción de la imagen. Esto es lo que se conoce como resolución espacial de la imagen.

2.2.8. IMÁGENES BLANCO/NEGRO Y COLOR

En definitiva, e independientemente del tipo de sensor utilizado, la imagen que ha de ser tratada por el computador se presenta digitalizada espacialmente en forma de matriz con una resolución de M x N elementos.

Si la imagen es en Blanco y Negro (B/N), se almacena un valor por cada píxel. Se suele utilizar un rango de valores para su representación, que generalmente es de 0 a 2n-1. Uno de los valores más utilizados de n es 8; esto significa que el rango de valores para este caso varía de 0 a 255. En este caso, el 0 representa el negro absoluto y el 255, el blanco absoluto. Esto indica que

podemos tener una resolución o precisión en los grises posibles de 256. El hecho de utilizar 256 niveles es porque con 8 bits del computador se pueden codificar 256 valores distintos desde la combinación 00000000, que representa el nivel 0, hasta la combinación 11111111, que representa el nivel 255.

En el caso de las imágenes en color, los elementos de la matriz vienen dados por tres valores, que representan cada uno de los componentes básicos del color en cuestión. Estos componentes son el Rojo (R), Verde (G) y Azul (B), el conocido código RGB. En este caso el conjunto de valores (0,0,0) es el negro absoluto; el (255,255,255), el blanco absoluto; el (255,0,0), el rojo puro; el (0,255,0), el verde puro; el (0,0,255), el azul puro. Como es lógico, la combinación de distintos valores proporciona otros colores, por ejemplo, el (255, 255, 51) es un tono de amarillo o el (204, 153, 102), es un tono marrón. El número de colores posible resulta ser 255.

2.2.9. RESOLUCIÓN ESPACIAL Y EN AMPLITUD

Para comprender mejor el sentido y la diferencia que existe entre resolución espacial y en amplitud, vamos a ilustrar los conceptos en las siguientes figuras:

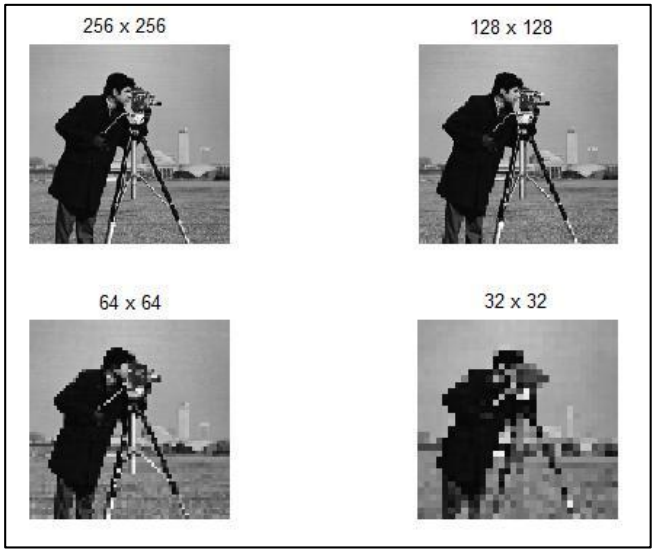


FIGURA 12: CUATRO REPRESENTACIONES DE LA MISMA IMAGEN CON VARIACIÓN EN EL NÚMERO DE PÍXELES UTILIZADOS

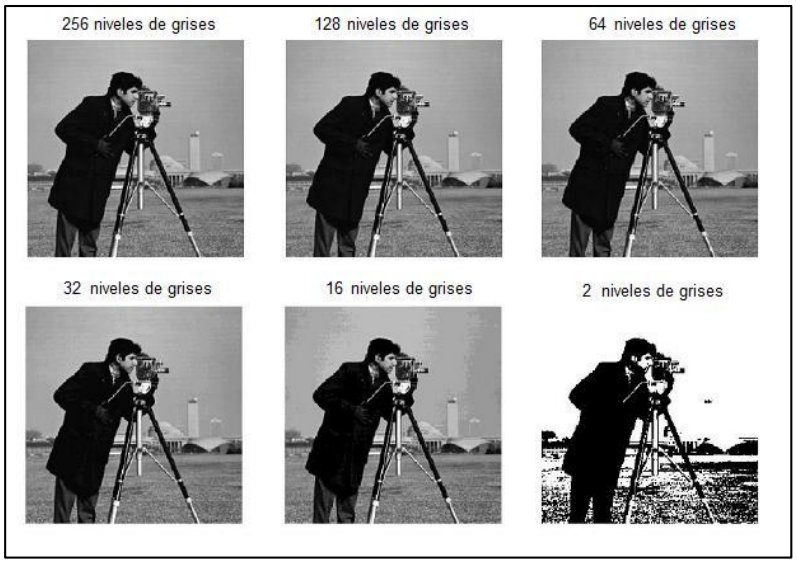


FIGURA 13: SEIS REPRESENTACIONES DE LA MISMA IMAGEN CON VARIACIÓN EN EL NÚMERO DE NIVELES DE GRIS UTILIZADOS

Como se muestra en las figuras anteriores, dependiendo del número de píxeles que tenga el dispositivo y de niveles de grises con que se trabaje en el computador, la imagen poseerá más o menos resolución espacial y en amplitud respectivamente.

2.2.10. REPRESENTACIÓN DE IMÁGENES DIGITALES

Como ya se ha mencionado antes, el termino imagen se refiere a una función de intensidad bidimensional, la cual puede ser representada como f(x,y),f(i,j),I(x,y),I(i,j), etc., donde x e y, o bien, i y j son las coordenadas espaciales y el valor de f o I en cualquier punto (x,y) o (i,j) es proporcional a la intensidad o nivel de gris de la imagen en ese punto.

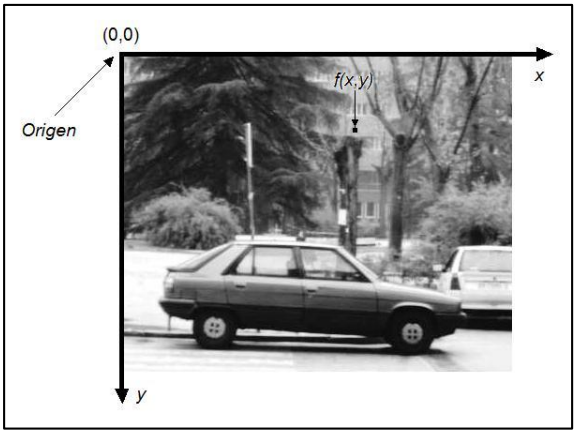


FIGURA 14: CONVENCIÓN DE EJES UTILIZADA PARA LA REPRESENTACIÓN DE IMÁGENES DIGITALES

2.3. PROCESAMIENTO Y ANALISIS DE IMÁGENES DIGITALES

Aunque la distinción entre procesamiento y análisis de imágenes digitales no es obvia de forma inmediata, el procesamiento de imágenes puede ser visto como una transformación de una imagen a otra imagen, es decir, a partir de una imagen, se obtiene otra imagen modificada. Por otro lado, el análisis es una transformación de una imagen en algo distinto a una imagen; en consecuencia, el análisis es un determinado tipo de información representando una descripción o una decisión. En la mayoría de los casos, las técnicas de análisis de imágenes digitales son aplicadas a imágenes que han sido procesadas previamente. Desde el punto de vista de un observador humano, el análisis de imágenes es una tarea fácil y rápida, algo que no ocurre en visión artificial. Igualmente, la capacidad de percepción humana permite procesar rápidamente una imagen para detectar en ella características de interés, por ejemplo, bordes o regiones.

2.4. PROCESAMIENTO BASICO DE IMÁGENES

El procesamiento de datos en el sistema de visión puede enfocarse desde 2 perspectivas:

Alteración píxel a píxel de los datos en una escala global (individuales).

Operaciones basadas en múltiples puntos (vecindad).

La generación de un nuevo píxel en una imagen será una función bien del valor de cada píxel en su localización individual, o bien de los valores de los píxeles en la vecindad de un píxel dado, como se indica en la figura siguiente:

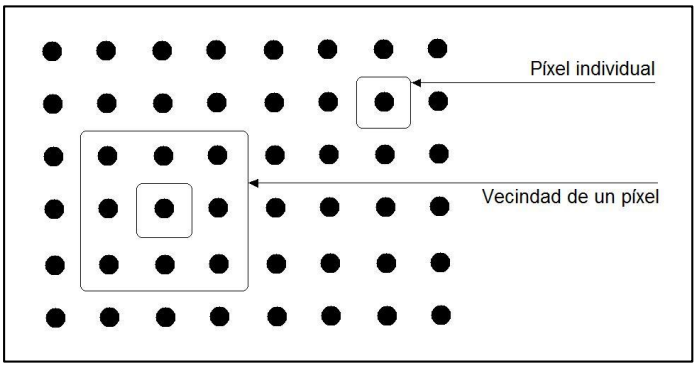


FIGURA 15: FUNCIONES DE PUNTO Y VECINDAD

Existen aún otras operaciones, que no se clasifican ni como individuales ni como vecindad ya que trasforman las imágenes por otros procedimientos; son transformaciones que operan globalmente sobre los valores de intensidad de la imagen cuyo efecto es un realzado de la imagen original, operaciones aritméticas y lógicas, estas últimas basadas en la teoría del algebra de Boole, y operaciones que realizan transformaciones geométricas, sin modificar los valores de intensidad.

2.4.1. OPERACIONES INDIVIDUALES

Las operaciones individuales implican la generación de una nueva imagen modificando el valor del pixel en una simple localización basándose en una regla global aplicada a cada localización de la imagen original. El proceso consiste en obtener el valor del píxel de una localización dada en la imagen, modificándolo por una operación lineal o no lineal y colocando el valor del nuevo pixel en la correspondiente localización de la imagen nueva. El proceso se repite para todas y cada de las localizaciones de los pixeles en la imagen original.

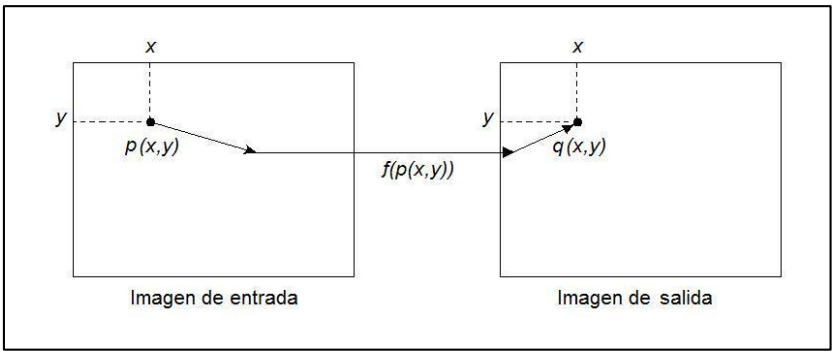


FIGURA 16: OPERACIÓN INDIVIDUAL

Como se aprecia en la figura anterior, el operador individual es una transformación uno a uno. El operador f se aplica a cada píxel en la imagen o sección de la imagen y la salida depende únicamente de la magnitud del correspondiente píxel de entrada; la salida es independiente de los píxeles adyacentes. La función transforma el valor del nivel de gris de cada píxel en la imagen y el nuevo valor se obtiene a través de la ecuación:

$$q(x,y) = f(p(x,y))$$

La función f puede ser un operador lineal o no lineal. El proceso matemático es relativamente simple. La imagen resultante es de la misma dimensión que la original.

2.4.2. OPERADOR IDENTIDAD

Este operador crea una imagen de salida que es idéntica a la imagen de entrada. La función de transformación es:

$$q=p$$

El operador identidad deja la imagen de entrada invariante. En la siguiente figura se muestra la función de transformación dada por la ecuación anterior:

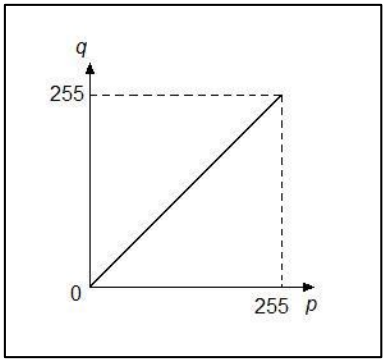


FIGURA 17: REPRESENTACIÓN DEL OPERADOR IDENTIDAD

2.4.3. OPERADOR INVERSO O NEGATIVO

Este operador crea una imagen de salida que es inversa de la imagen de entrada. Este operador es útil en diversas aplicaciones tales como imágenes médicas. Para una imagen con valores de gris en el rango de 0 a 255 la función de transformación resulta ser:

$$q = 255 - p$$

En la siguiente figura se muestra la función de transformación dada por la ecuación anterior:

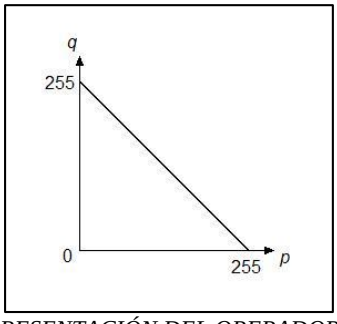


FIGURA 18: REPRESENTACIÓN DEL OPERADOR INVERSO

2.4.4. OPERADOR UMBRAL

Esta clase de transformación crea una imagen de salida binaria a partir de una imagen de grises, donde el nivel de transición está dado por el parámetro de entrada p_1. La función de transformación es la siguiente:

$$q = \begin{cases} 0 & para\ p \leq p_1 \\ 255 & para\ p > p_1 \end{cases}$$

En la siguiente figura se muestra la función de transformación dada por la ecuación anterior:

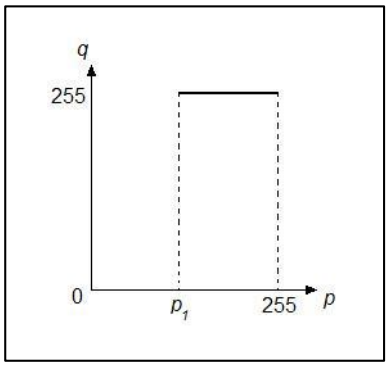


FIGURA 19: REPRESENTACIÓN DEL OPERADOR UMBRAL

2.4.5. TRANSFORMACIÓN DE VECINDAD

En las operaciones de vecindad modifican el nuevo valor del píxel en la imagen de salida depende de una combinación de los valores de los píxeles en la vecindad de la imagen original que está siendo transformada.

Dentro de la categoría de operaciones de vecindad se incluyen las operaciones de filtrado. Las operaciones de filtrado tienen la particularidad de eliminar un determinado rango de frecuencias de las imágenes.

2.4.5.1. Nociones y propiedades de vecindad

Se dice que todo píxel p, de coordenadas (x,y), tiene cuatro píxeles que establecen con él una relación de vecindad horizontal o vertical, que son:

Horizontal: (x-1, y) y (x+1, y) Vertical: (x, y-1) y (x, y+1)

Estos cuatro pixeles definen lo que se conoce como entorno de vecindad-4 y nos referimos a ellos como E_4 (p).

Los cuatro vecinos diagonales de p tienen coordenadas:

(x-1, y-1), (x+1, y-1), (x-1, y+1), (x+1, y+1)

y nos referimos a ellos como E_D (p). Estos píxeles junto con los E_4 (p) se llaman los vecinos-8 de p y se denotan como E_8 (p).

Existen excepciones dadas cuando el píxel (x, y) es un punto del borde de la imagen, en cuyo caso algunos de los vecinos definidos anteriormente no existen.

2.4.5.2. Conectividad

Sea V el conjunto de valores de intensidad de los píxeles que se permiten estén adyacentes, por ejemplo, si sólo se desea que exista conectividad entre los píxeles con intensidades 80,81 y 83, entonces V = {80,81,83}. Consideremos tres tipos básicos de conectividad:

Conectividad-4. Dos píxeles p y q con valores de V están 4-conectados si q está en el conjunto E_4 (p).

Conectividad-8. Dos píxeles p y q con valores de V están 8-conectados si q está en el conjunto E_8 (p).

Conectividad-m (mixta). Dos píxeles p y q con valores de V están m-conectados si q está en el conjunto E_4 (p) o q está en E_D (p) y E_4 (p)∩E_4 (q)=∅.

Un píxel p es contiguo a otro píxel q si están conectados. Se puede definir la adyacencia-4, 8 o m, dependiendo del tipo de conectividad especificada. Dos subconjuntos imagen S1 y S2 son contiguos si algún píxel se S1 es contiguo a algún píxel de S2.

Un camino desde el píxel p con coordenadas (x, y) hasta un píxel q con coordenadas (s, t) es una secuencia de varios píxeles con coordenadas,

$$(x\_0,y\_0),(x\_1,y\_1),\ldots,(x\_n, y\_n)$$

donde $(x\_0, y\_0) = (x, y)$ y $(s, t),(x\_i, y\_i)$ son adyacentes a $(x\_(i-1), y\_(i-1))$, con $1 \leq i \leq n$, y n es la longitud del camino. Se pueden definir caminos -4, 8 o m dependiendo del tipo de adyacencia usada.

Si p y q son píxeles de un subconjunto de imagen S, entonces p está conectado a q en S si existe un camino desde p hasta q formado de píxeles pertenecientes a S. Dado un píxel p cualquiera de S, el conjunto de píxeles de S que están conectados a p se llaman componente conectado de S. Se deduce que dos píxeles cualesquiera de un componente conectado están a su vez conectados entre sí y que los componentes conectados distintos son disjuntos.

Un camino simple es un camino sin píxeles repetidos y un camino cerrado es un camino simple en el cual el primer píxel es un vecino del último.

2.5. OPERACIONES DE FILTRADO

Las operaciones de filtrado basan su operatividad en la Convolución de la imagen utilizando el denominado núcleo de Convolución. Cabe distinguir dos tipos de filtros: paso alto y paso bajo, que en el contexto de la teoría de señales supone que los primeros dejan pasar las altas frecuencias de la señal y los segundos, las bajas. En el caso de las imágenes nos referimos a frecuencias espaciales. De una forma, las altas frecuencias se asocian a cambios bruscos de

intensidad en pequeños intervalos espaciales, es decir, bordes, mientras que las bajas frecuencias se refieren a cambios lentos en la intensidad.

2.5.1. Filtros Paso Bajo

En la siguiente figura se muestran algunos núcleos de convolución que caracterizan los filtros paso bajo:

$$PB_1 \equiv \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$PB_2 \equiv \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$PB_3 \equiv \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

FIGURA 20: TRES NÚCLEOS REPRESENTATIVOS DE FILTROS PASO BAJO

2.5.2. Filtros Paso Alto

En la siguiente figura se muestran algunos núcleos de Convolución que caracterizan los filtros paso alto:

$$PA_1 \equiv \begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

$$PA_2 \equiv \begin{bmatrix}-1 & -1 & -1\\ -1 & 9 & -1\\ -1 & -1 & -1\end{bmatrix}$$

$$PA_3 \equiv \begin{bmatrix}1 & -2 & 1\\ -2 & 5 & -2\\ 1 & -2 & 1\end{bmatrix}$$

FIGURA 21: TRES NÚCLEOS REPRESENTATIVOS DE FILTROS PASO ALTO

2.6. HISTOGRAMA

El histograma de una imagen es la representación gráfica o analítica de la distribución relativa de cada valor posible de pixel de imagen, y en caso de imágenes grises de 8 bits será un vector de 256 componentes, siendo la componente i el número de pixeles de nivel i en la imagen, dividido por el número total de pixeles:

El histograma es formalmente la función estadística de densidad de probabilidad en forma discreta de los distintos niveles de gris dentro de la imagen.

$$histograma[i] = \frac{N_{pixeles\ \ de\ \ nivel\ \ i}}{N_{total}} \qquad 0 \leq i \leq 255$$

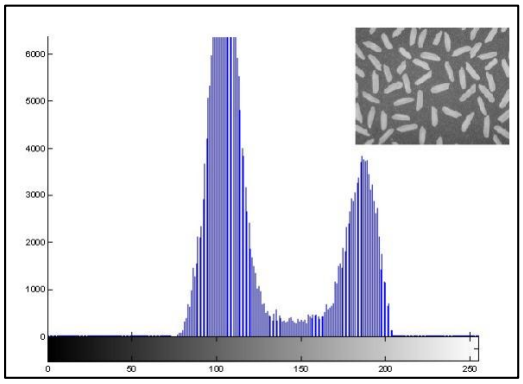


FIGURA 22: HISTOGRAMA

2.7. RECONOCIMIENTO DE PATRONES[6]

El reconocimiento de patrones es la ciencia que se ocupa de los procesos sobre ingeniería, computación y matemáticas relacionados con objetos físicos o abstractos, con el propósito de extraer información que permita establecer propiedades de entre conjuntos de dichos objetos.

El reconocimiento de patrones, también llamado lectura de patrones, identificación de figuras y reconocimiento de formas, consiste en el

[6] https://es.wikipedia.org/wiki/Reconocimiento_de_patrones

reconocimiento de patrones de señales. Los patrones se obtienen a partir de los procesos de segmentación, extracción de características y descripción donde cada objeto queda representado por una colección de descriptores. El sistema de reconocimiento debe asignar a cada objeto su categoría o clase (conjunto de entidades que comparten alguna característica que las diferencia del resto). Para poder reconocer los patrones se siguen los siguientes procesos:

- adquisición de datos
- extracción de características
- toma de decisiones

El punto esencial del reconocimiento de patrones es la clasificación: se quiere clasificar una señal dependiendo de sus características. Señales, características y clases pueden ser de cualquiera forma, por ejemplo, se puede clasificar imágenes digitales de letras en las clases «A» a «Z» dependiendo de sus píxeles o se puede clasificar ruidos de cantos de los pájaros en clases de órdenes aviares dependiendo de las frecuencias.

2.7.1. SENSOR

El sensor es el dispositivo encargado de la adquisición de datos. Ha de ser capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas. Las variables de instrumentación dependen del tipo de sensor y pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, etc.

2.7.2. EXTRACCIÓN DE CARACTERÍSTICAS

Es el proceso de generar características que puedan ser usadas en el proceso de clasificación de los datos. En ocasiones viene precedido por un preprocesado de la señal, necesario para corregir posibles deficiencias en los datos debido a errores del sensor, o bien para preparar los datos de cara a posteriores procesos en las etapas de extracción de características o clasificación.

Las características elementales están explícitamente presentes en los datos adquiridos y pueden ser pasados directamente a la etapa de clasificación. Las características de alto orden son derivadas de las elementales y son generadas por manipulaciones o transformaciones en los datos.

2.7.3. CLASIFICACIÓN

La clasificación trata de asignar las diferentes partes del vector de características a grupos o clases, basándose en las características extraídas. En esta etapa se usa lo que se conoce como aprendizaje automático, cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender.

2.8. CONCEPTO DE OPENCV[7]

OpenCV es una biblioteca multiplataforma que permite desarrollar aplicaciones de visión por ordenador en tiempo real. Se centra en el procesamiento de imágenes, captura de vídeo y análisis, incluyendo características como detección de rostros y detección de objetos.

OpenCV significa Open Source Computer Vision Library; por lo tanto, es una librería de tratamiento de imágenes, destinada principalmente a aplicaciones de visión por computador en tiempo real. Una de las ventajas principales es que puede funcionar en muchas plataformas, existen versiones para Windows, Linux y MacOs.

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel; desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OSX y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas

[7] https://www.tutorialspoint.com/opencv/

en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

FIGURA 23: OPENCV

2.8.1. Características principales

- Optimizado para procesamiento de imágenes en tiempo real y aplicaciones de visión artificial
- La interfaz primaria de OpenCV está en C + +
- También hay interfaces C, Python y JAVA completas
- Las aplicaciones OpenCV se ejecutan en Windows, Android, Linux, Mac e iOS
- Optimizado para procesadores Intel

2.8.2. Módulos OpenCV

OpenCV tiene una estructura modular. A continuación, se enumeran los módulos principales de OpenCV.

Núcleo

Este es el módulo básico de OpenCV. Incluye estructuras de datos básicas (por ejemplo, estructura de datos Mat) y funciones básicas de procesamiento de imágenes. Este módulo también es ampliamente utilizado por otros módulos como highgui, etc.

Highgui

Este módulo ofrece funciones sencillas de interfaz de usuario, varios códecs de imagen y vídeo, capacidades de captura de imagen y video, manipulación de ventanas de imagen, manejo de barras de seguimiento y eventos de ratón y etc. Si desea capacidades de interfaz de usuario avanzadas, debe utilizar marcos de interfaz de usuario como Qt, WinForms, etc, por ejemplo, cargar y mostrar imágenes, capturar vídeo desde un archivo o cámara, escribir imagen y vídeo en el archivo

Imgproc

Este módulo incluye algoritmos básicos de procesamiento de imágenes incluyendo filtrado de imágenes, transformaciones de imagen, conversiones de espacio de color y etc.

Vídeo

Este es un módulo de análisis de video que incluye algoritmos de seguimiento de objetos, algoritmos de substracción de fondo y etc.

Objdetect

Esto incluye algoritmos de detección y reconocimiento de objetos para objetos estándar.

OpenCV ahora se utiliza extensivamente para desarrollar el procesamiento avanzado de la imagen y las aplicaciones de la visión de la computadora. Ha sido una herramienta para estudiantes, ingenieros e investigadores en todos los rincones del mundo.

2.9. GLOSARIO DE TÉRMINOS BÁSICOS

OCR: El reconocimiento óptico de caracteres (ROC), generalmente conocido como reconocimiento de caracteres y expresado con frecuencia con la sigla OCR (del inglés Optical Character Recognition), es un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos. Así podremos interactuar con estos

mediante un programa de edición de texto o similar. En los últimos años la digitalización de la información (textos, imágenes, sonido, etcétera) ha devenido un punto de interés para la sociedad. En el caso concreto de los textos, existen y se generan continuamente grandes cantidades de información escrita, tipográfica o manuscrita en todo tipo de soportes. En este contexto, poder automatizar la introducción de caracteres evitando la entrada por teclado implica un importante ahorro de recursos humanos y un aumento de la productividad, al mismo tiempo que se mantiene, o hasta se mejora, la calidad de muchos servicios.

ZONNING: Zonificación en OCR es el proceso de creación de zonas que corresponden a atributos específicos de un elemento de página. Una zona se puede identificar como un gráfico sin texto, alfanumérico o numérico. Las aplicaciones de OCR generalmente permiten la zonificación automática o manual.

CLASIFICACIÓN DE IMAGEN: La clasificación de imagen hace referencia a la tarea de extraer clases de información de una imagen. El ráster resultante de la clasificación de imagen se puede utilizar para crear mapas temáticos. Dependiendo de la interacción entre el analista y el equipo durante la clasificación, existen dos tipos de clasificación: supervisada y no supervisada.

SISTEMA: Un sistema es un conjunto ordenado de elementos que se encuentran insurreccionados entre sí para lograr un fin en común.

CLASIFICACIÓN SUPERVISADA: La clasificación supervisada utiliza firmas espectrales obtenidas de las muestras de capacitación para clasificar una imagen. Con la ayuda de la barra de herramientas Clasificación de imagen, puede crear fácilmente muestras de capacitación para representar las clases que desea extraer. También puede crear con facilidad un archivo de firma a partir de las muestras de capacitación que, a continuación, las herramientas de clasificación multivariante utilizarán para clasificar la imagen.

CLASIFICACIÓN NO SUPERVISADA: La clasificación sin supervisión busca clases espectrales (o clústeres) en una imagen multibanda sin la intervención del analista. La barra de herramientas Clasificación de imagen ayuda a realizar la clasificación sin supervisión proporcionando acceso a las herramientas para

crear clústeres, capacidad para analizar la calidad de los clústeres y acceso a las herramientas de clasificación.

BLOB ANÁLISIS: Un algoritmo utilizado en visión artificial que identifica objetos segmentados y los mide según parámetros morfométricos (tamaño, diámetro, perímetro, etc.) o densitométicos (nivel de gris medio, media, color)

ATRIBUTO: Un atributo es una especificación que define una propiedad de un Objeto, elemento o archivo. También puede referirse o establecer el valor específico para una instancia determinada de los mismos.

CCD: es la abreviatura de Charge-Coupled Device. Un sensor CCD es un dispositivo semiconductor sensible a la luz, que convierte las partículas de luz (fotones) en cargas eléctricas (electrones). Las cámaras CCD son una de los dos tipos de cámaras que dominan el mercado de la visión, conjuntamente con las cámaras CMOS.

CMOS: Es el acrónimo de Complementary Metal Oxide Semiconductor. Esta tecnología funciona como un fotodiodo donde la luz genera una corriente que es representativo de la cantidad de luz que impacta en cada píxel, por tanto difiere significativamente de la tecnología CCD. Hay un número de ventajas en la utilización de los sensores CMOS con respecto a los CCD entre los que se destaca el coste, velocidad, anti-blooming, y características de respuesta programable (Ej. Respuesta con múltiples pendientes).

ESCALA DE GRISES: En visión este término se refiere a una imagen monocroma con una gradación de grises. Una cámara monocroma que trabaje a 8 bits generará una imagen con 256 tonos de gris. Si la cámara trabaja a 12 bits su escala de gises será de 4096 niveles.

HISTOGRAMA: Es una representación gráfica de los valores de todos los píxeles de la imagen. Generalmente a la izquierda de la gráfica se representa el valor cero correspondiente al negro y a la derecha se representa el valor máximo (que depende del número de bits) correspondiente al blanco. La curva del histograma representa la cantidad de píxeles de cada valor de gris que hay en la imagen. Si la imagen es en color se acostumbran a representar los histogramas de valores de rojo, verde y azul.

MODELO: Un modelo es una representación de un objeto, sistema o idea, de forma diferente al de la entidad misma. El propósito de los modelos es ayudarnos a explicar, entender o mejorar un sistema. Un modelo de un objeto puede ser una réplica exacta de éste o una abstracción de las propiedades dominantes del objeto.

INTELIGENCIA ARTIFICIAL: Es aquella inteligencia exhibida por artefactos creados por humanos (es decir, artificial). A menudo se aplica hipotéticamente a los computadores. El nombre también se usa para referirse al campo de la investigación científica que intenta acercarse a la creación de tales sistemas.

ROBOTICA: Es la rama de la tecnología que se dedica al diseño, construcción, operación, disposición estructural, manufactura y aplicación de los robots. La robótica combina diversas disciplinas como son: la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física. Otras áreas importantes en robótica son el álgebra, los autómatas programables, la animatrónica y las máquinas de estados.

VISION ARTIFICIAL: La visión artificial, también conocida como visión por computador (del inglés computer vision) o visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen. (La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes, por ejemplo, caras humanas).

SIMILITUD: Relación entre personas o cosas que tiene características comunes.

2.10. HIPÓTESIS

Utilizando visión artificial, se reconocerá automáticamente el número del documento nacional de identificación de una persona.

SISTEMA DE RECONOCIMIENTO AUTOMATICO DE NUMEROS UTILIZANDO SIMILARIDAD DEL COSENO PARA VISIÓN ARTIFICIAL

III. MARCO METODOLÓGICO

3.1. MÉTODOS Y PROCEDIMIENTOS

Para el desarrollo del trabajo de tesis se hizo uso del siguiente material:

- 1 computadora personal de 1.2 GHz, 4GB RAM, 500GB DD.
- Una webcam de 30 fps, resolución de 640 x 480 pixeles.

Se utilizó como sistema operativo, Windows 10.

Como lenguaje de programación se utilizó Microsoft Visual C.Net. Además, se hizo uso de las librerías de Visión Artificial, OpenCV, estas librerías son software libre y además que se pueden utilizar en C++, tienen una funcionalidad y flexibilidad fácil de utilizar. Por otro lado, Visual C es un potente lenguaje de programación que da velocidad de procesamiento.

Para llegar al objetivo planteado, se llevó a cabo el siguiente procedimiento:

3.1.1. ACONDICONAMIENTO DE LA ESCENA

Para la implementación del sistema, se imprimieron en papel con fondo blanco, los números del 0 al 9, con diferentes tipos de letra y en negrita. También se preparó números hechos a mano para comprobar el funcionamiento del sistema implementado. La distancia de la imagen a la cámara webcam fue de 20cm.

En cuanto a la iluminación, se tomó en cuenta la luz natural ambiental y en caso de ausencia de esta, se necesita luz artificial, que puede ser con leds, es decir no se ha tomado en cuenta la oscuridad.

3.1.2. CAPTURA DE LA IMAGEN

Para capturar la escena de la imagen, se hace uso de una cámara webcam. Tan luego es expuesta la hoja con los números respectivos, se hace la captura en color, utilizando el modelo RGB. La imagen de captura tiene un ancho de 640 pixeles por una altura de 480 pixeles.

3.1.3. CONVERSIÓN A ESCALA DE GRISES

Como sólo hay objeto números y fondo, trabajar con imágenes en color es algo innecesario. Por lo tanto, para pasar a la siguiente etapa es necesario convertir la imagen RGB a imagen escala de grises, es decir, de 3 componentes por pixel, solo nos quedamos con una, así se logra procesar menos información. Esta conversión se logra promediando las 3 componentes.

3.1.4. FILTRADO PASA BAJO

El filtrado pasa bajo se hace con la finalidad de eliminar ruido presente por la cámara misma. Como el ruido no afecta al objeto de los números en cuanto a forma, de todas maneras, se utiliza un filtro promediador, cuyo kernel es [1 1 1, 1 1 1, 1 1 1].

Este kernel o núcleo se desliza por toda la imagen, teniendo como resultado un suavizado de esta.

3.1.5. BINARIZADO

Una vez que hemos suavizado la imagen, separamos objeto y fondo. A los objetos que son los números les asignamos el color blanco y al fondo le asignamos color negro. Este proceso de quedarnos con una imagen en blanco y negro se llama binarización. El umbral que hemos tenido en cuenta es de 120, pixeles con valores menores a este umbral le asignamos 0 ó negro y pixeles mayores a este valor le asignamos 255, o color blanco. El valor del umbral 120 fue obtenido mediante pruebas. La función que se utiliza es threshold.

3.1.6. ETIQUETADO DE OBJETOS

Para separar cada objeto número hay que etiquetarlos. La forma de etiquetar cada objeto número es mediante una función de las librerías opencv, findContours. Esta función me devuelve una identificación individual de pixeles vecinos y conectados de cada objeto número. Además, devuelve el área de cada

objeto número, como sabemos el área de cada objeto está dado por el número de pixeles presente en ese objeto.

3.1.7. SEGMENTACIÓN DE ACUERDO CON EL AREA

Ahora que tenemos etiquetados cada componente y además su área. Podemos filtrar el tamaño de cada objeto para la siguiente etapa. Podemos decir que pixeles mayores a un mínimo y pixeles menores a un máximo van a ser reconocidos. Con esto evitamos que pixeles de objetos pequeños, producidos por el ruido, no entren a la etapa de reconocimiento.

3.1.8. RECONOCIMIENTO DEL NÚMERO

La etapa del reconocimiento se lleva a cabo, comparando las imágenes almacenadas y las imágenes capturadas.

Para esto se leen previamente las imágenes almacenadas en formato de archivo bmp. Se tienen 5 modelos diferentes de cada número y tienen un tamaño de 15 x 25 pixeles de ancho y altura respectivamente.

Cuando el programa implementado lee los pixeles de cada número, los agrupa y promedia en pixeles de 5 x 5. Como tenemos objetos de números de 15 pixeles de ancho, resultan 3 valores de ancho, y como tenemos 25 pixeles de altura tenemos, 5 valores de altura, es decir en total tenemos 15 valores por cada objeto número. A este proceso se le denomina zonning. En lugar de trabajar con imágenes de 15 x 25 que son 375 pixeles, se trabaja con 15 valores (3 x 5) que pertenecen al promediado de cada subregión de 5 x 5 de cada objeto número. Ahora tenemos 15 valores por cada objeto número en lugar de 375 pixeles. Hay menos datos que representan la misma información. En la captura de la imagen que se desea procesar, se obtienen exactamente igual los 15 valores, pero previo procesamiento descrito anteriormente, que son los pasos del 3.1.1 al 3.1.7.

Entonces, como ya tenemos los 15 valores de cada objeto número de las imágenes archivadas y los 15 valores del objeto número de la imagen capturada a procesar, el siguiente paso es comparar los 15 valores de la imagen capturada con cada uno de los 15 valores de las imágenes almacenadas. A una de ellas se

debe corresponder y quien hace esta comparación es la fórmula de Similaridad del coseno. Si el resultado que se obtiene después de aplicar la fórmula es cercano a 0, quiere decir que las imágenes de los números comparados no son iguales y si el resultado obtenido es cercano a 1 quiere decir que se trata de dos imágenes del mismo número. Por lo tanto, se trabaja con un umbral de 0.8, para decir si es mayor a éste el número a buscar o ser reconocido, es muy parecido al almacenado.

IV. RESULTADOS Y DISCUSIÓN

4.1. RESULTADOS

Después de aplicar la metodología explicada en el capítulo anterior, pasamos a probar los algoritmos realizados.

Primero, mostramos los resultados de las imágenes con números individuales, luego mostramos los resultados de las imágenes con números agrupados, impresos en computadora y por último mostramos los resultados de imágenes con números escritos a mano.

Para cada captura, se presentan las imágenes binarizadas, previamente capturadas y luego el resultado del reconocimiento.

Se presentan las imágenes de números impresos y generados en computadora en el software procesador de textos, Microsoft Word.

En cada figura se puede observar como el sistema reconoce cada número, y en la parte superior izquierda, el valor de la similitud del coseno obtenido al número que se parece.

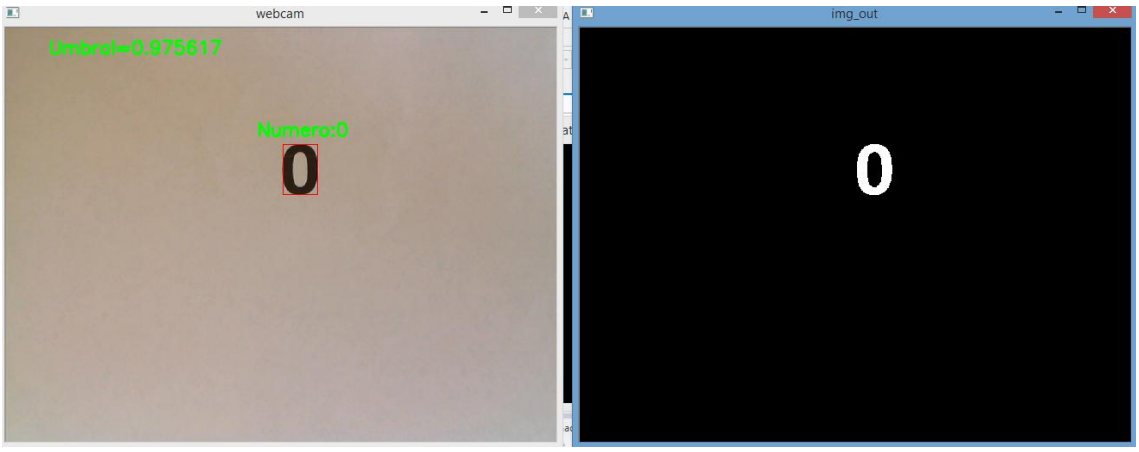


FIGURA 24 Imagen del número 0 reconocido y su respectivo procesamiento

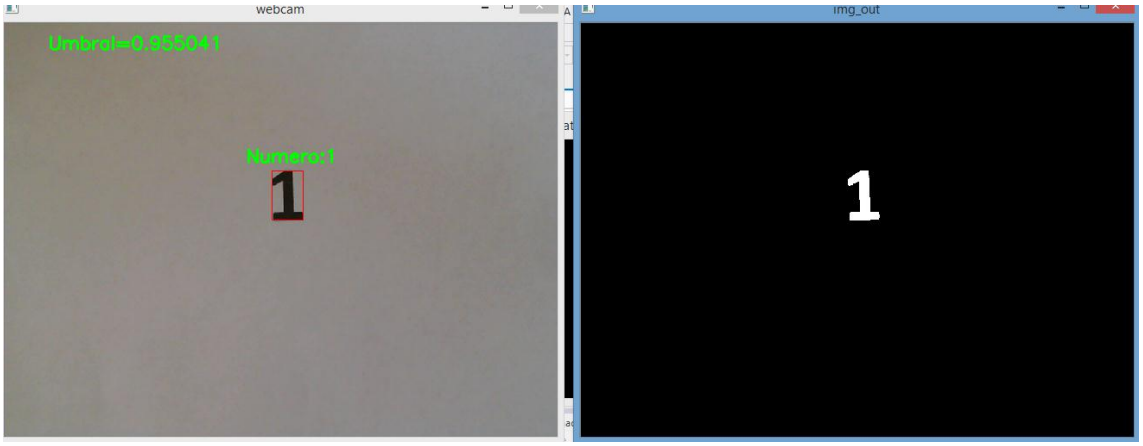


FIGURA 25 *Imagen del número 1 reconocido y su respectivo procesamiento*

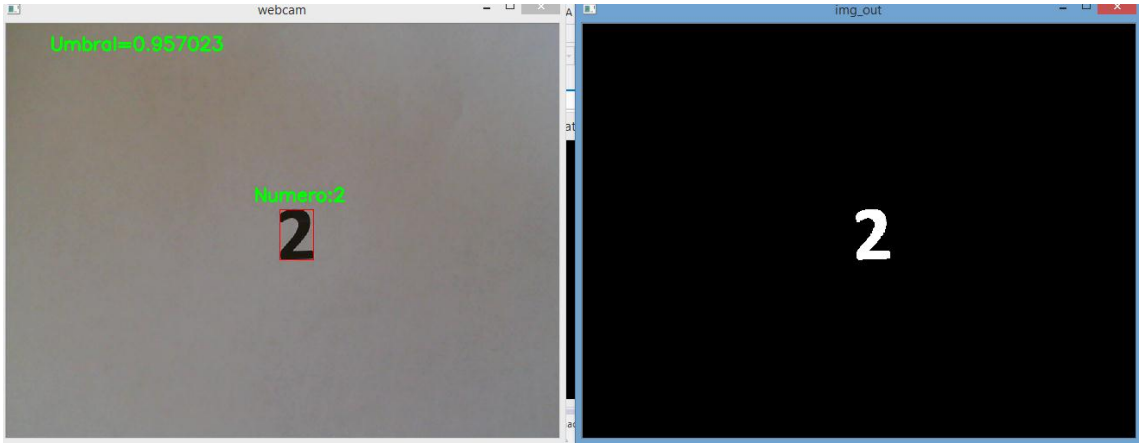


FIGURA 26 *Imagen del número 2 reconocido y su respectivo procesamiento*

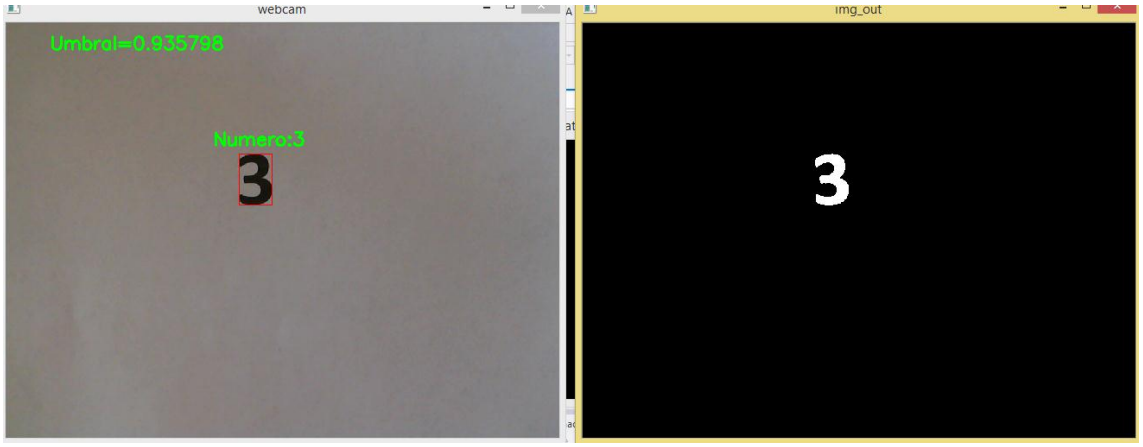


FIGURA 27 *Imagen del número 3 reconocido y su respectivo procesamiento*

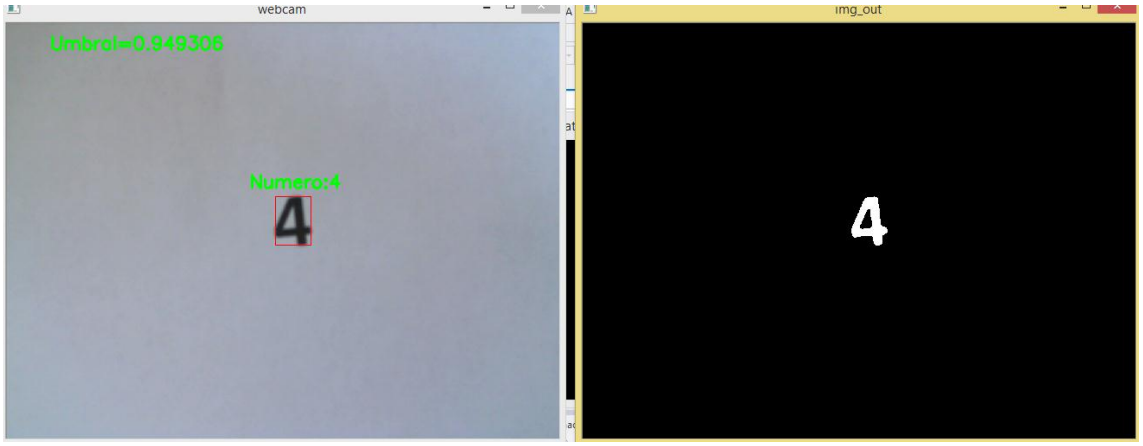


FIGURA 28 *Imagen del número 4 reconocido y su respectivo procesamiento*

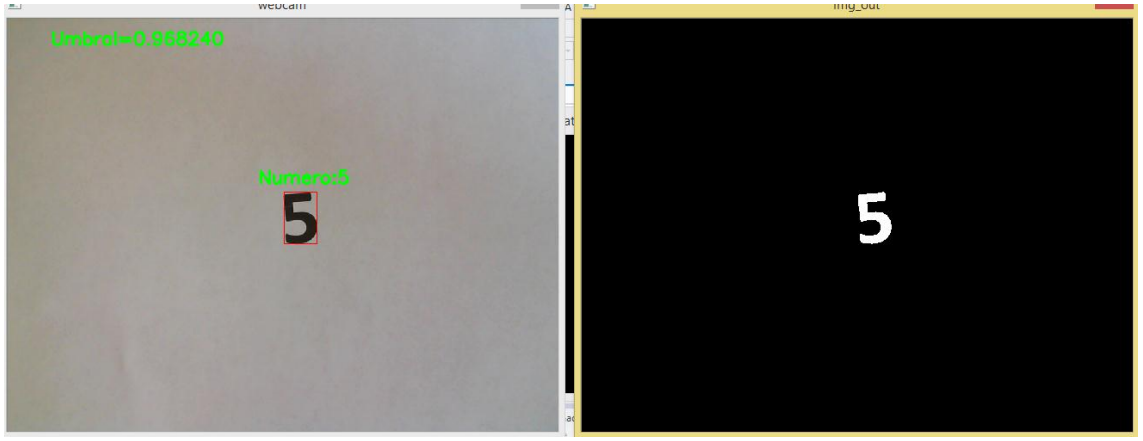


FIGURA 29 *Imagen del número 5 reconocido y su respectivo procesamiento*

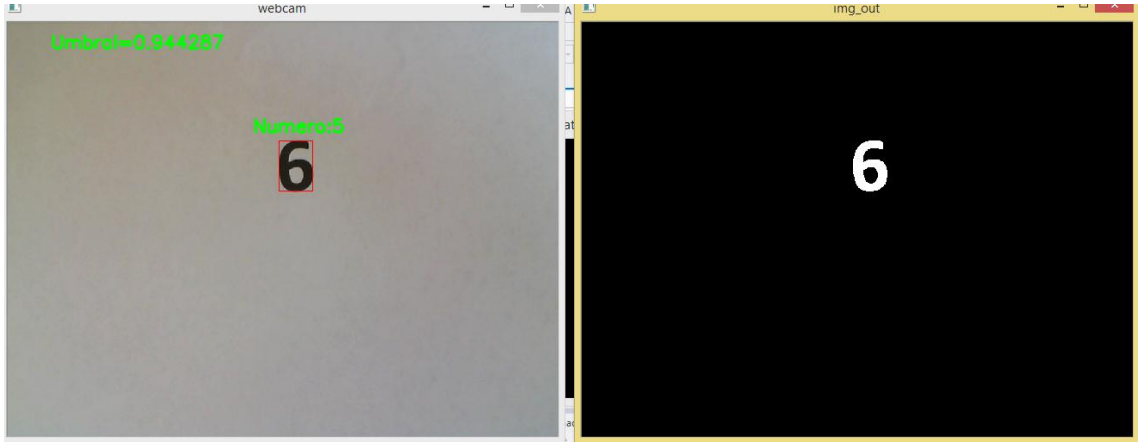


FIGURA 30 *Imagen del número 6 reconocido y su respectivo procesamiento*

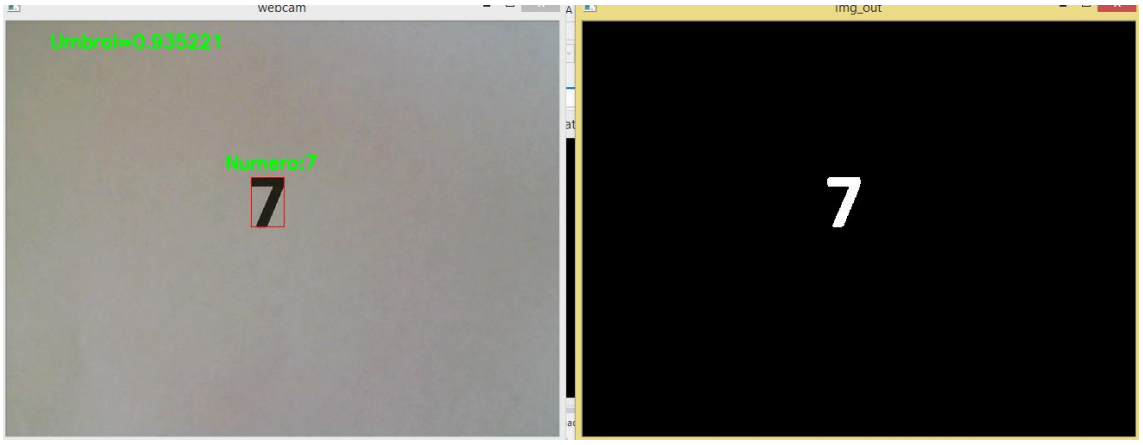


FIGURA 31 *Imagen del número 7 reconocido y su respectivo procesamiento*

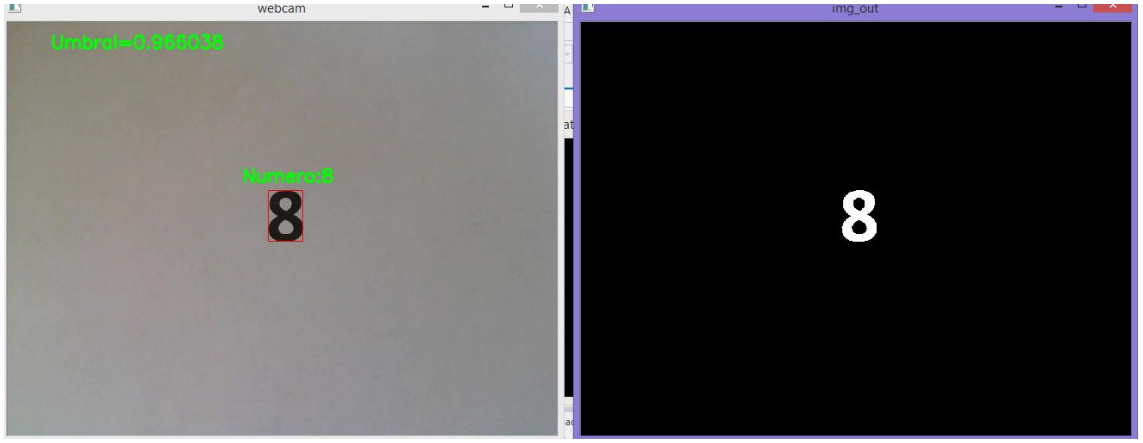


FIGURA 32 *Imagen del número 8 reconocido y su respectivo procesamiento*

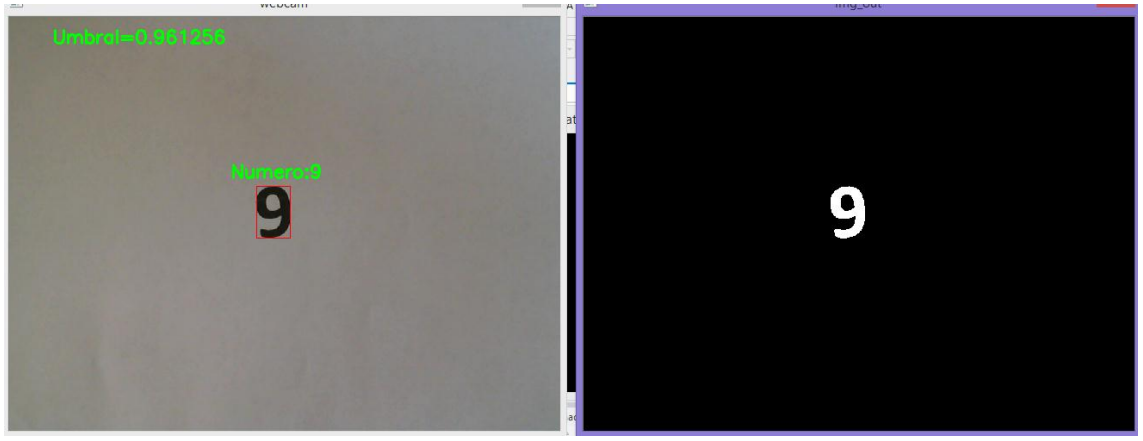


FIGURA 33 *Imagen del número 9 reconocido y su respectivo procesamiento*

Ahora, se presentan los resultados de las imágenes de números hechos a mano:

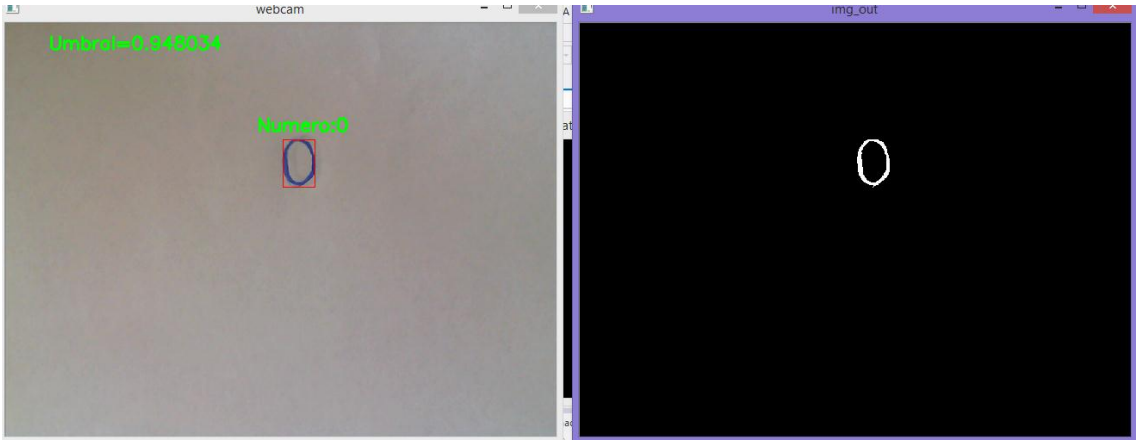


FIGURA 34 *Imagen del número 0 a mano reconocido y su respectivo procesamiento*

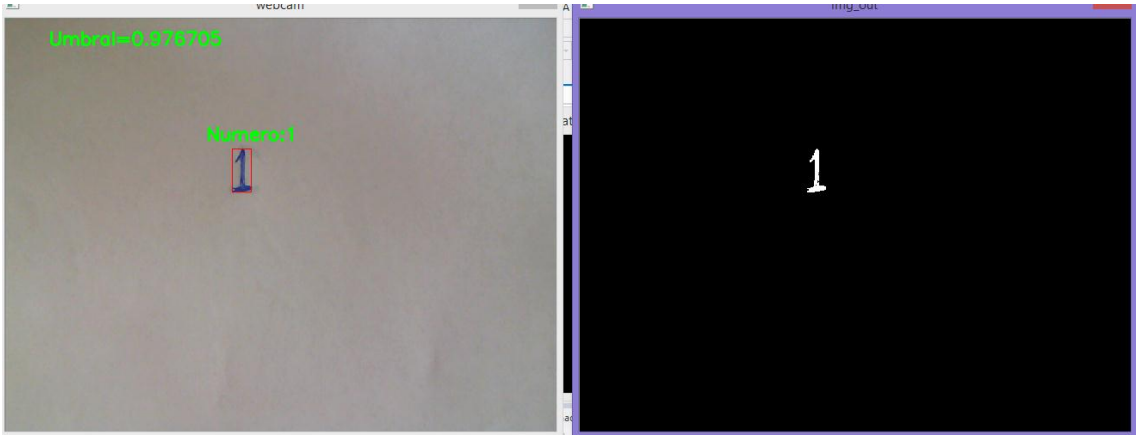


FIGURA 35 *Imagen del número 1 a mano reconocido y su respectivo procesamiento*

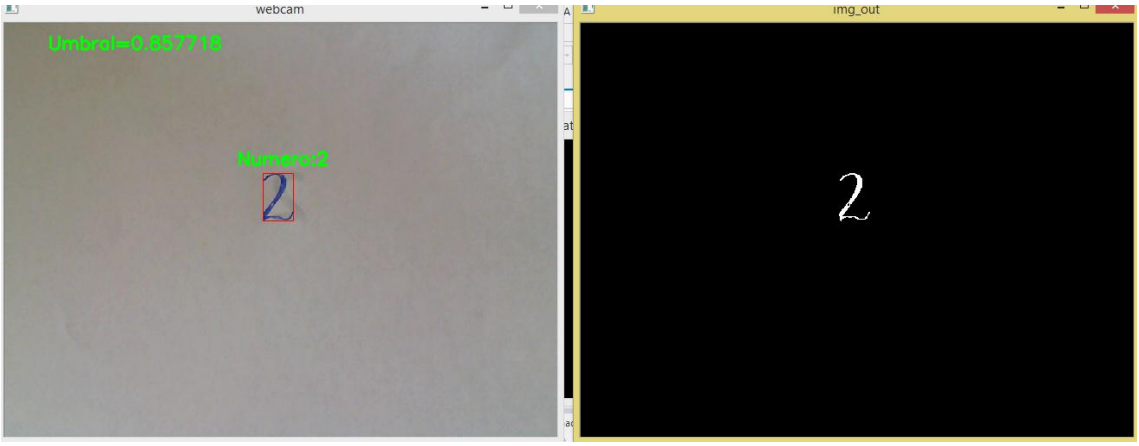


FIGURA 36 *Imagen del número 2 a mano reconocido y su respectivo procesamiento*

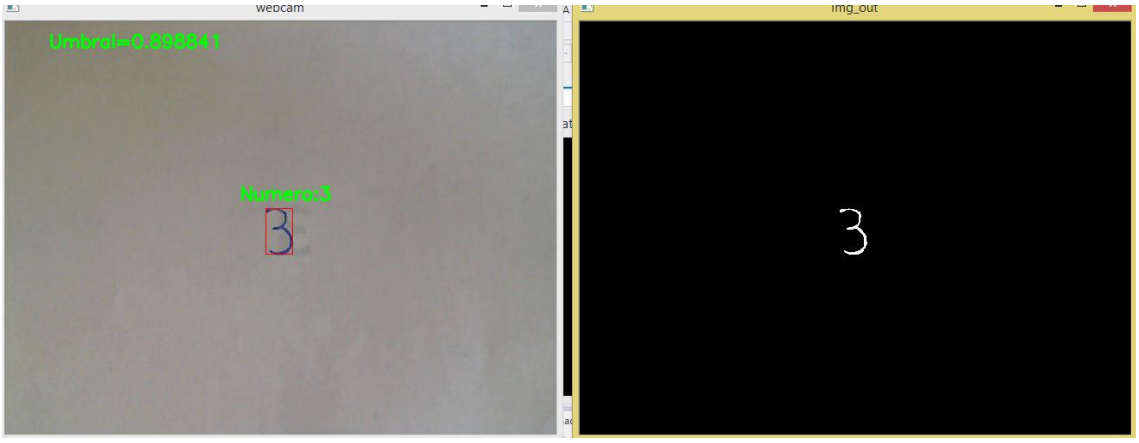


FIGURA 37 *Imagen del número 3 a mano reconocido y su respectivo procesamiento*

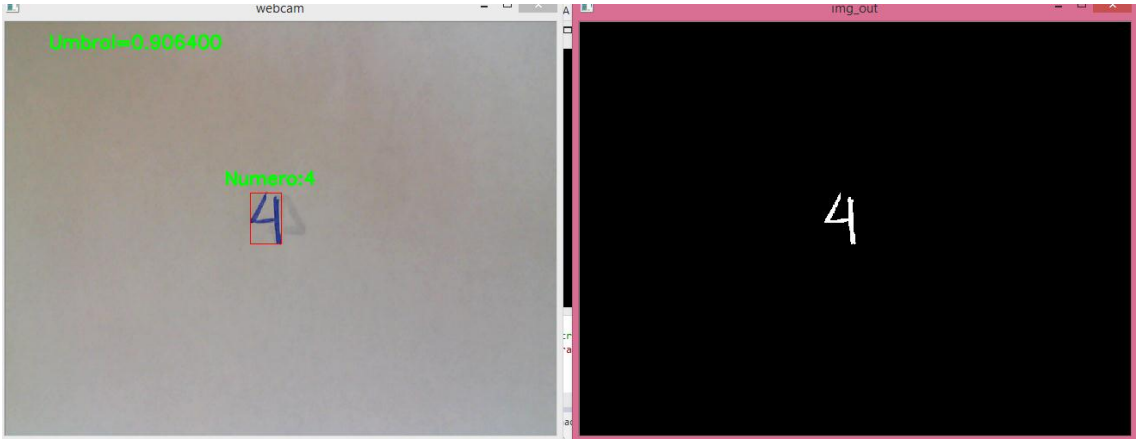


FIGURA 38 *Imagen del número 4 a mano reconocido y su respectivo procesamiento*

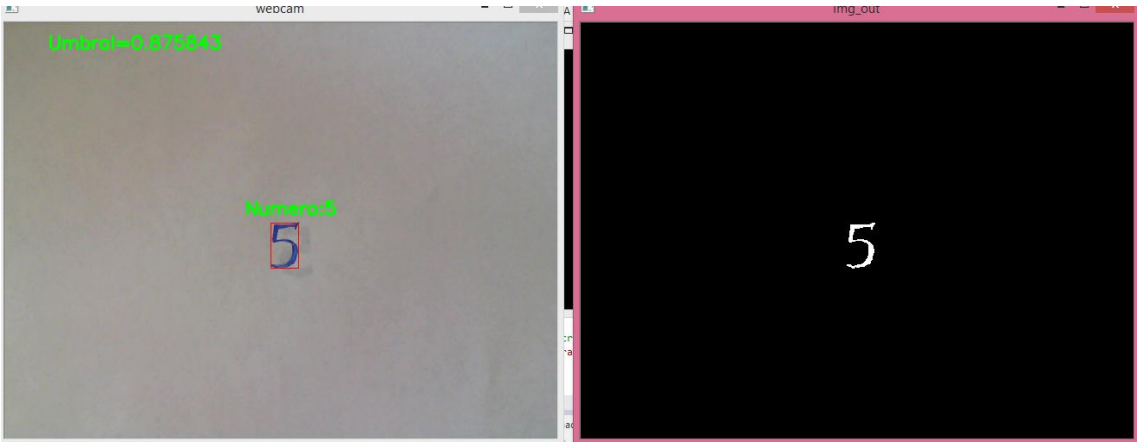


FIGURA 39 *Imagen del número 5 a mano reconocido y su respectivo procesamiento*

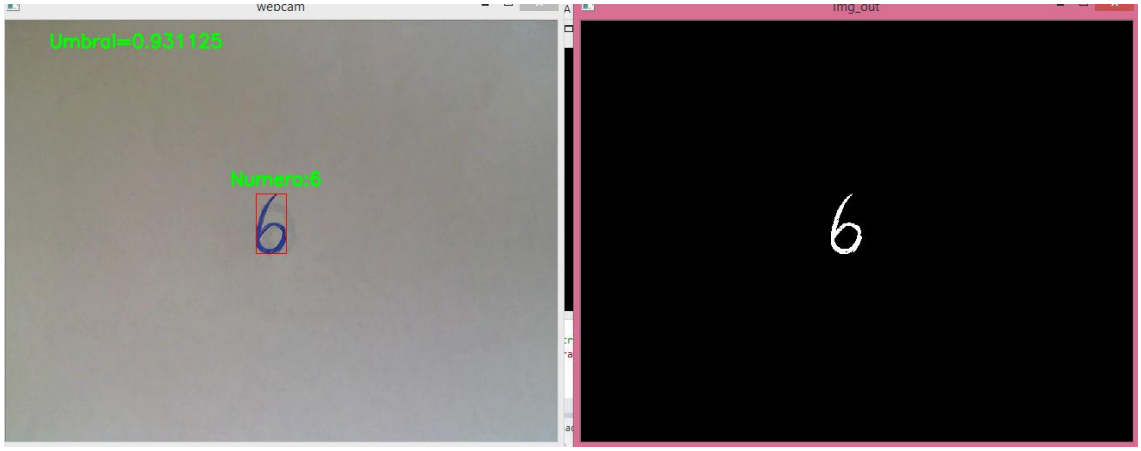


FIGURA 40 Imagen del número 6 a mano reconocido y su respectivo procesamiento

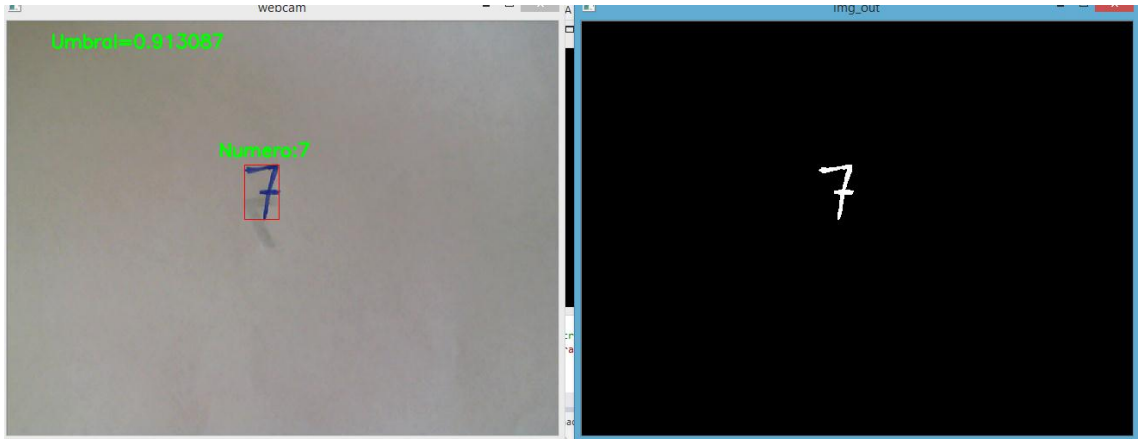


FIGURA 41 Imagen del número 7 a mano reconocido y su respectivo procesamiento

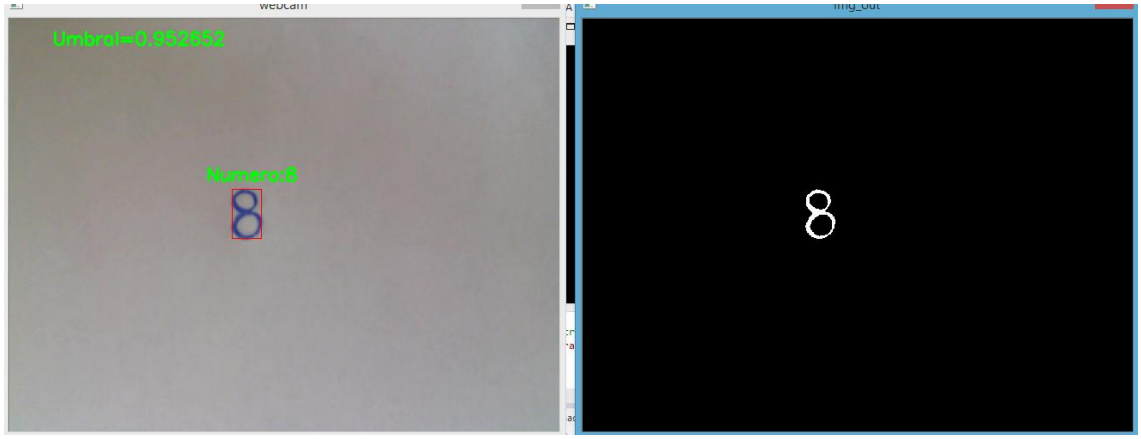


FIGURA 42 Imagen del número 8 a mano reconocido y su respectivo procesamiento

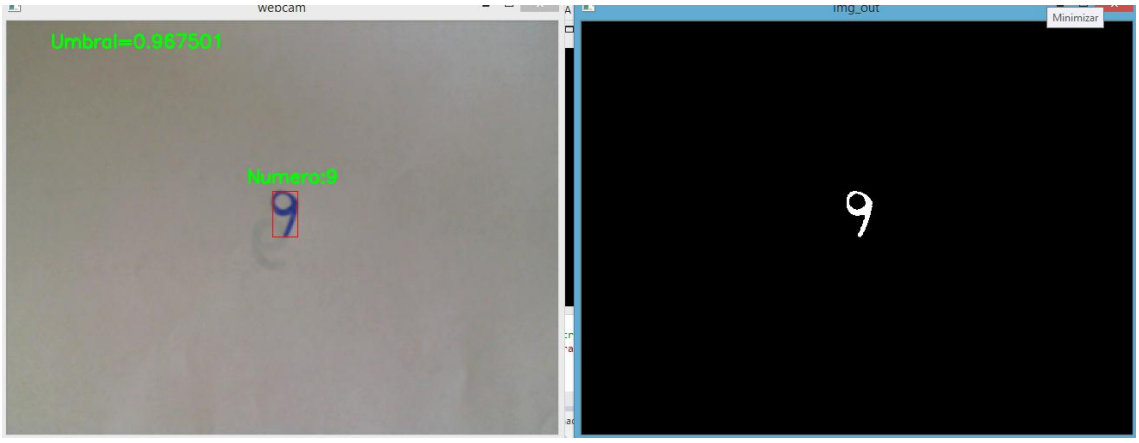


FIGURA 43 Imagen del número 9 a mano reconocido y su respectivo procesamiento

Número	Cos(ang):Num_comp	Cos(ang): Num_mano
0	0.975617	0.948034
1	0.955041	0.976705
2	0.957023	0.857717
3	0.935798	0.898841
4	0.949306	0.906400
5	0.968240	0.875843
6	0.944287	0.931125
7	0.935221	0.913087
8	0.966038	0.952652
9	0.961256	0.967501

Tabla 1 Resultados del reconocimiento de los números

Se pueden observar los resultados en la tabla 1. En la primera columna están los números a reconocer, en la segunda están los resultados de la similitud del coseno de imágenes en computadora y en la tercera columna, están los resultados de la similitud del coseno de imágenes hechas a mano.

A continuación, se presenta el resultado de las imágenes con números impresos generados en computadora en el software procesador de textos, Microsoft Word. Esta vez se presentan, conjunto de números, como imagen de entrada.

FIGURA 44 Procesamiento de la imagen 1 de entrada, Binarizada

FIGURA 45 Resultado del reconocimiento de imagen 1

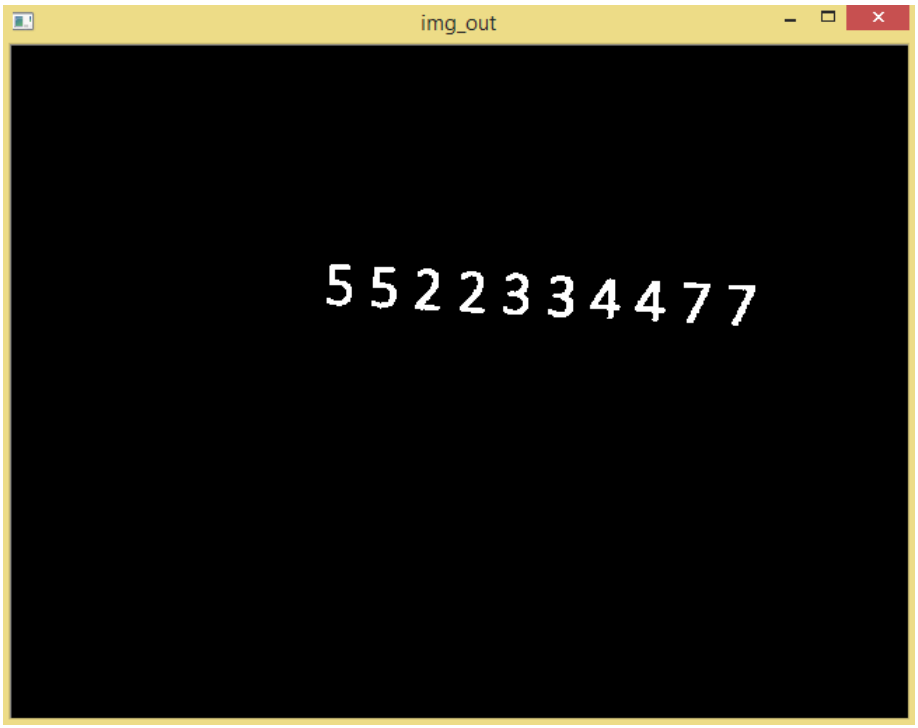


FIGURA 46 *Procesamiento de la imagen 2 de entrada, Binarizada*

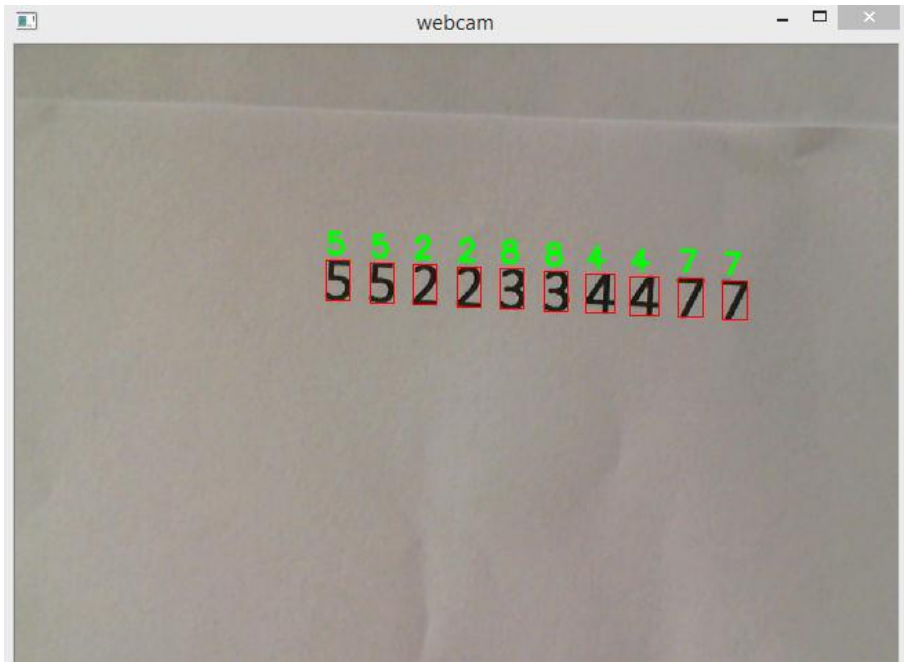


FIGURA 47 *Resultado del reconocimiento de imagen 2*

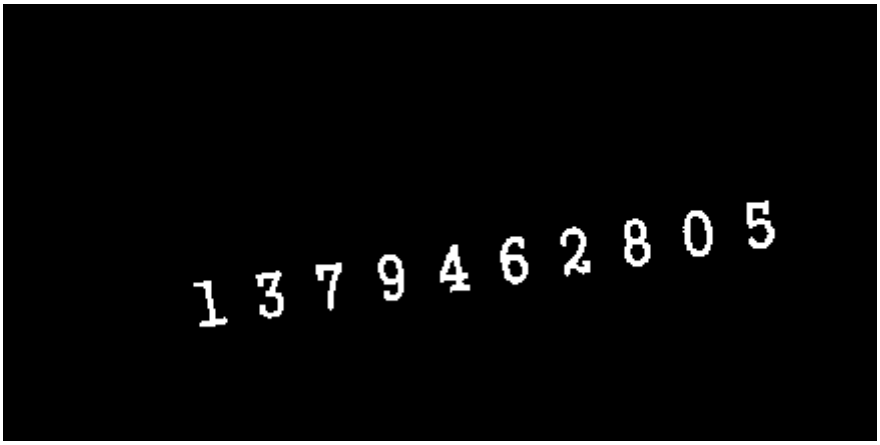


FIGURA 48 Procesamiento de la imagen 3 de entrada, Binarizada

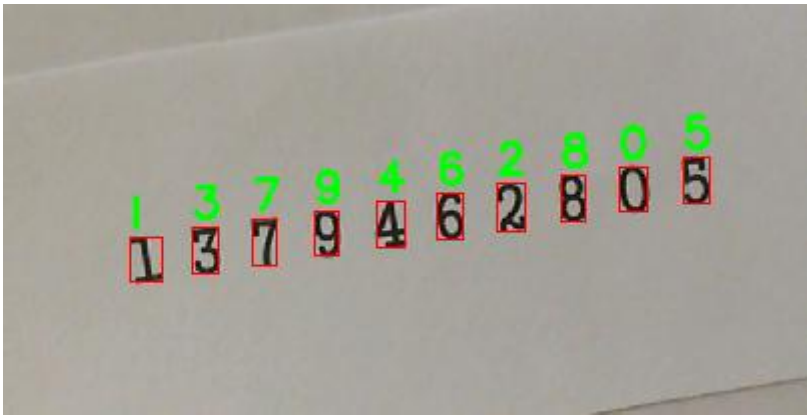


FIGURA 49 Resultado del reconocimiento de imagen 3

A continuación, se observa los resultados del procesamiento de imágenes de un conjunto de números hechos a mano.

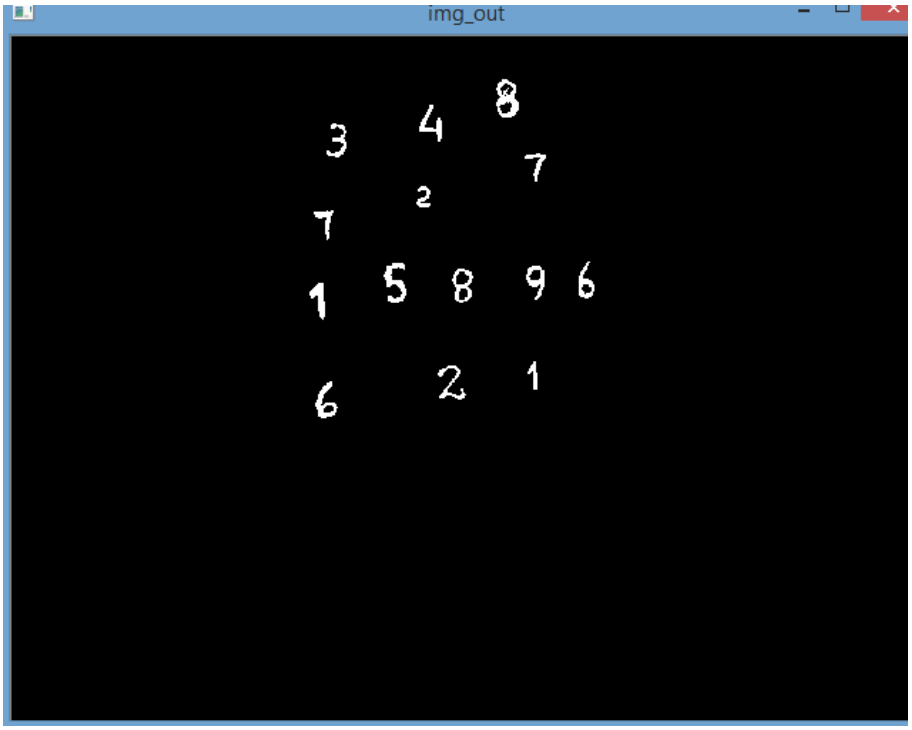


FIGURA 50 *Procesamiento de la imagen 4 de entrada, Binarizada*

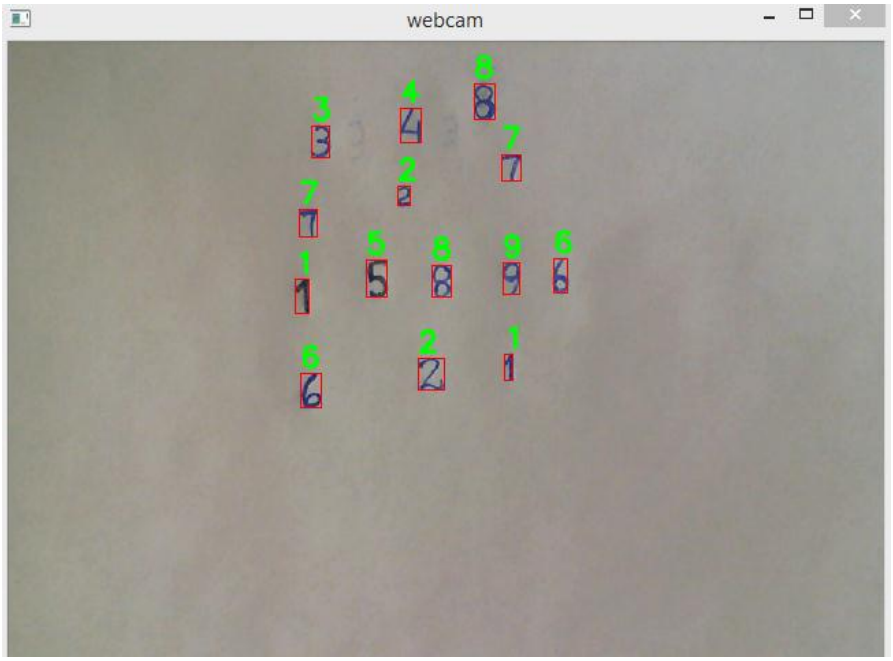


FIGURA 51 *Resultado del reconocimiento de imagen 4*

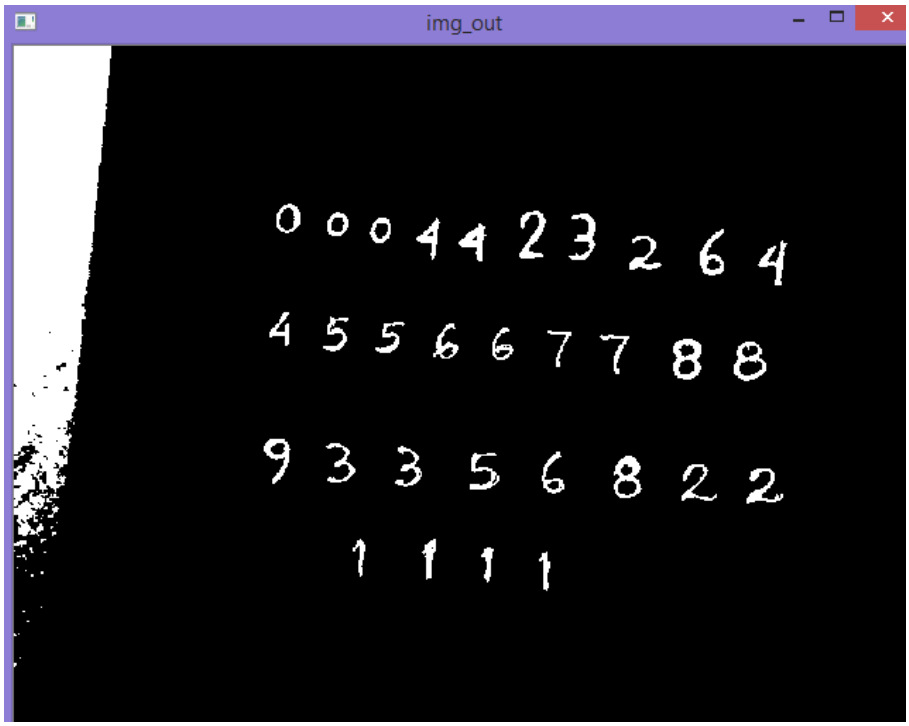


FIGURA 52 Procesamiento de la imagen 5 de entrada, Binarizada

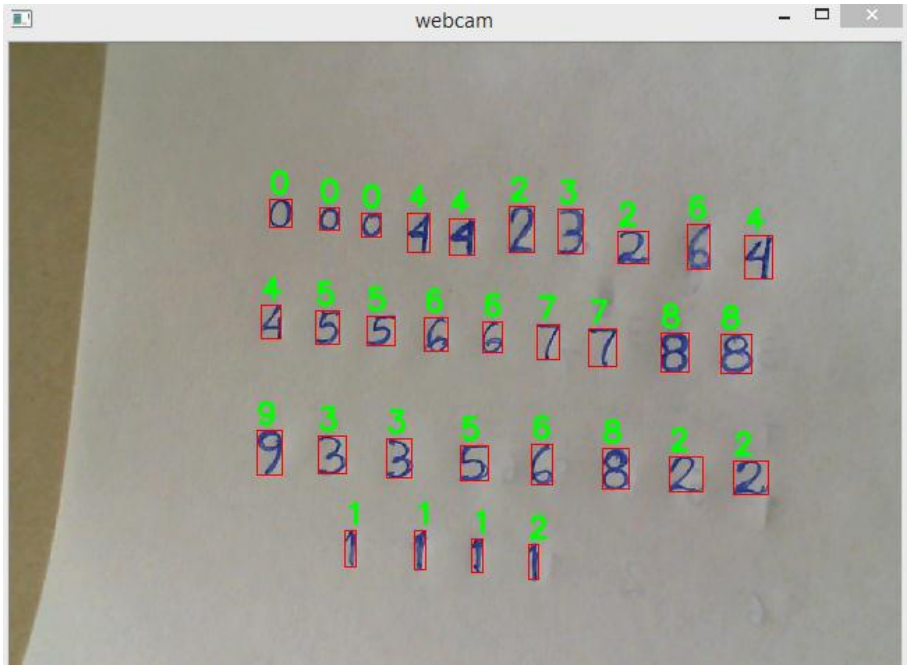


FIGURA 53 Resultado del reconocimiento de imagen 5

4.2. DISCUSIÓN

La metodología empleada es óptima, ya que se puede ver en las imágenes binarizadas, el resultado desde la adquisición en RGB, conversión a escala de grises, filtrado pasa bajo hasta la binarización.

Luego viene el etiquetado y el reconocimiento. En el etiquetado se puede observar la delimitación con un rectángulo de color rojo. Por encima de este rectángulo vemos el número reconocido.

Se puede observar en la tabla 1, los resultados del reconocimiento de cada número, en la segunda columna se presentan los números hechos en computadora y en la tercera columna, los resultados de los números hechos a mano.

Casi todos los resultados son mayores a 0.9, lo que quiere decir que son casi iguales a sus respectivos números, ya que como dijimos si se esperaba un resultado cercano a 1, se trataban de números iguales y si el valor resultado de la similitud del coseno era 0, se trataban de imágenes totalmente diferentes.

En el segundo grupo de imágenes, donde las entradas son imágenes que contienen varios números, se da un reconocimiento también del 100%.

Se puede observar, más que todo, en las imágenes con números hechos a mano, no siempre son iguales, tienen un margen de aleatoriedad, los números de entrada no siempre son iguales, y aun así el sistema los reconoce.

CONCLUSIONES

Se puede concluir que el sistema de reconocimiento automático de números utilizando similaridad del coseno para visión artificial ha sido implementado con éxito. La similaridad del coseno, sirve como sistema de comparación y de medición de números.

El sistema reconoce tipo de letra impresa hechas en computadora y a mano. Cada número ha sido reconocido con una similitud mayor a 0.85.

Los números han sido reconocidos incluso cuando hay poco contraste entre los caracteres y el fondo.

Los números han sido reconocidos aun cuando existe una variación significativa en el ancho y la altura.

Cabe mencionar que el sistema fue pensado para introducir números hechos en computadora, mas no, para números hechos a mano, pero se puede ver la robustez del sistema.

Los números han sido reconocidos cuando las letras están inclinadas y distorsionadas.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C++ optimizados, aprovechando además las capacidades que proveen los procesadores multi-núcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

RECOMENDACIONES

Para implementar el sistema se han utilizado por cada número, un conjunto de 5 diferentes plantillas, almacenadas en forma de archivo. Esto puede influir a la hora del reconocimiento, es decir que, si se tienen más modelos de plantillas por cada número, el sistema puede funcionar mucho mejor.

Cada persona tiene una forma diferente de escribir los números, estos modelos diferentes pueden ser ingresados para ser almacenados como plantillas.

REFERENCIAS BIBLIOGRÁFICAS

1. Rafael C. Gonzales, Richard E. Woods, Tratamiento Digital de Imágenes. USA, Addison Wesley Iberoamericana, S. A. 1996. 755 pp.
2. Michael Alder, An Introduccion to Pattern Recognition, Mike Alder, 2001.
3. Abraham Kandel, Horst Bunke, Mark Last, Applied graph theory in computer visión and pattern recognition, Springer-Verlag Berlin Heidelberg, 2007.
4. Bernd Jahne, Digital image processing, Springer-Verlag Berlin Heidelberg, 2002.
5. Gonzalo Pajares Martinsanz, Jesús Manuel de la Cruz García, José Manuel molina pascual, Juan Cuadrado Pardo, Alejandro López Correa, "Imágenes digitales - Procesamiento practico con java, Alfa Omega Grupo Editor S.A. 2004
6. Computer vision with the OpenCV library; Gary Bradsky y Adrian Kaebler.
7. Manual original proporcionado por intel: Intel® Open Source Computer
8. Vision Library.
9. Learning OpenCV: Computer Vision with the OpenCV Library, Bradski, G.,
10. Kaebler, A.

ANEXOS

ANEXO 1: RESULTADO DE MAS IMÁGENES DE ENTRADA AL SISTEMA

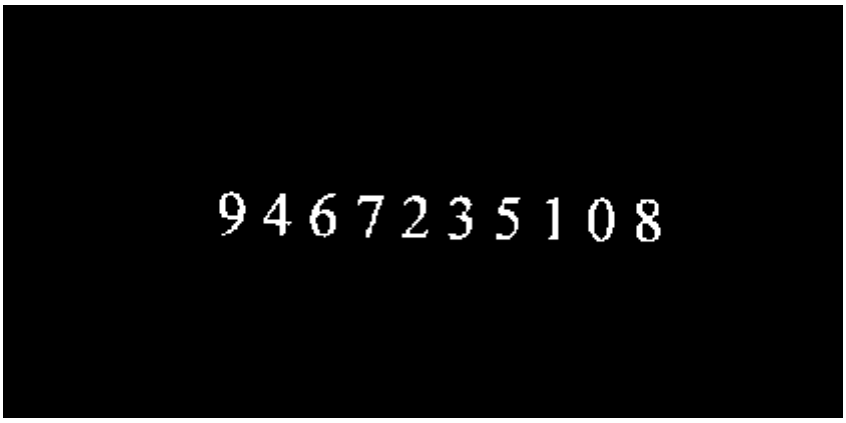


FIGURA 54: Imagen resultada del procesamiento, binarización

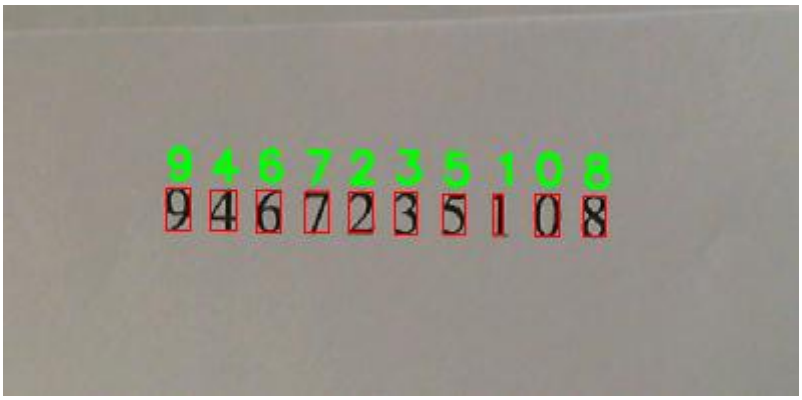


FIGURA 55: Imagen que muestra el reconocimiento

FIGURA 56: Imagen resultada del procesamiento, binarización

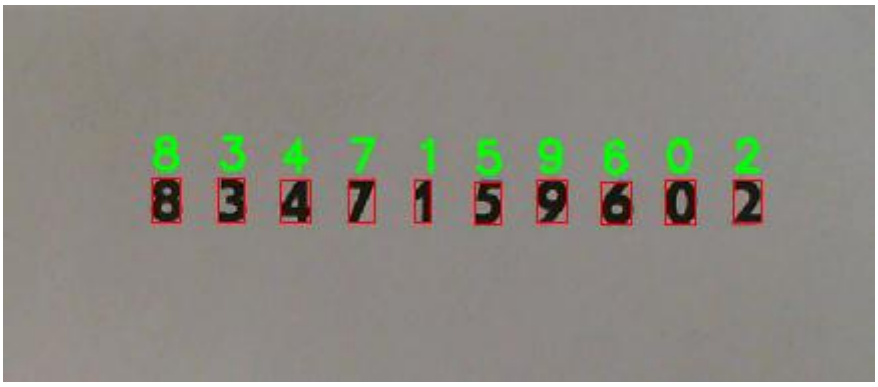


FIGURA 57: Imagen que muestra el reconocimiento

FIGURA 58: Imagen resultada del procesamiento, binarización

FIGURA 59: Imagen que muestra el reconocimiento

En la instalación se utiliza OpenCV 2.4.5 y Visual Studio 2012. Pero es casi lo mismo para otras versiones de OpenCV y Visual Studio.

Hay 2 maneras de instalar OpenCV en la computadora. La forma más adecuada es la instalación mediante el uso de las bibliotecas preconstruidas. Por lo tanto, voy a discutir cómo instalar OpenCV mediante bibliotecas preconstruidas.

En primer lugar debe tener un IDE adecuado, instalado Visual Studio Express 2012 que es una edición gratuita. (Registrarse dando la dirección de correo electrónico para obtener la clave de producto gratuita).

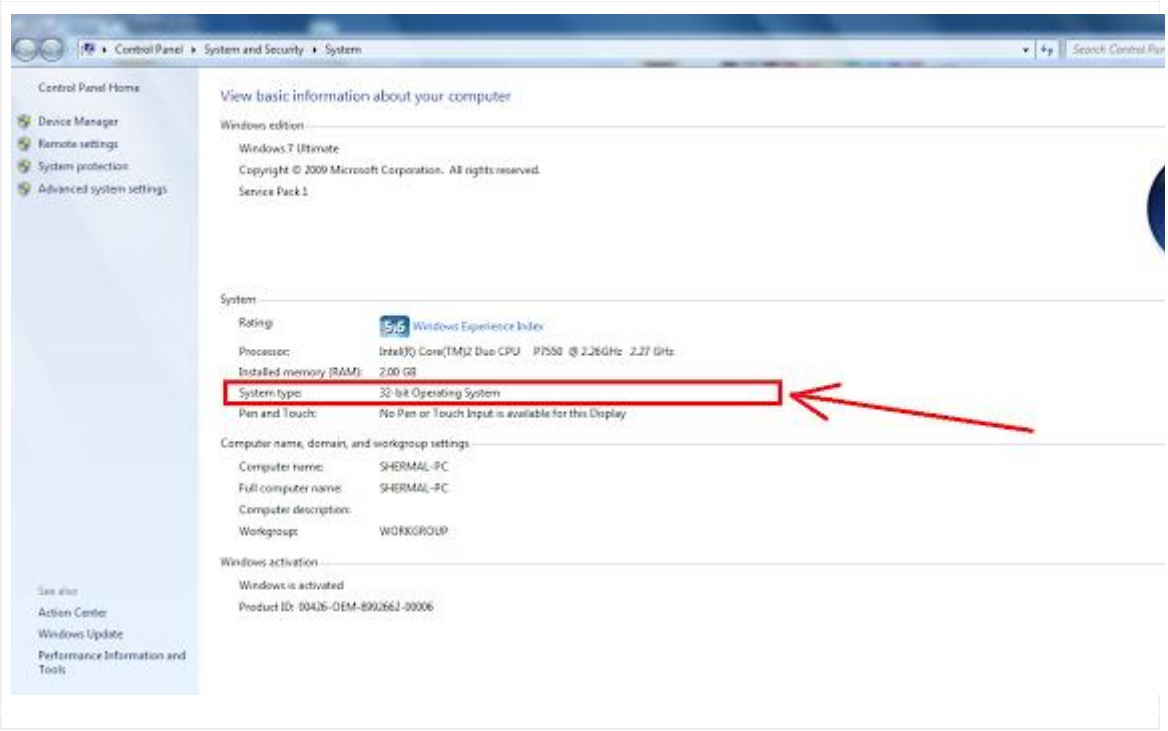
Descargar OpenCV y elegir la versión OpenCV 2.4.5.

A continuación, haga doble clic en el archivo descargado "OpenCV-2.4.5.exe".

Ahora configurar correctamente las variables de entorno para poder usar OpenCV.

Pasos para configurar la variable de entorno

- Hacer clic derecho en '**Mi PC**' y luego en '**Propiedades**' en el menú desplegable. Se verá una ventana como esta.



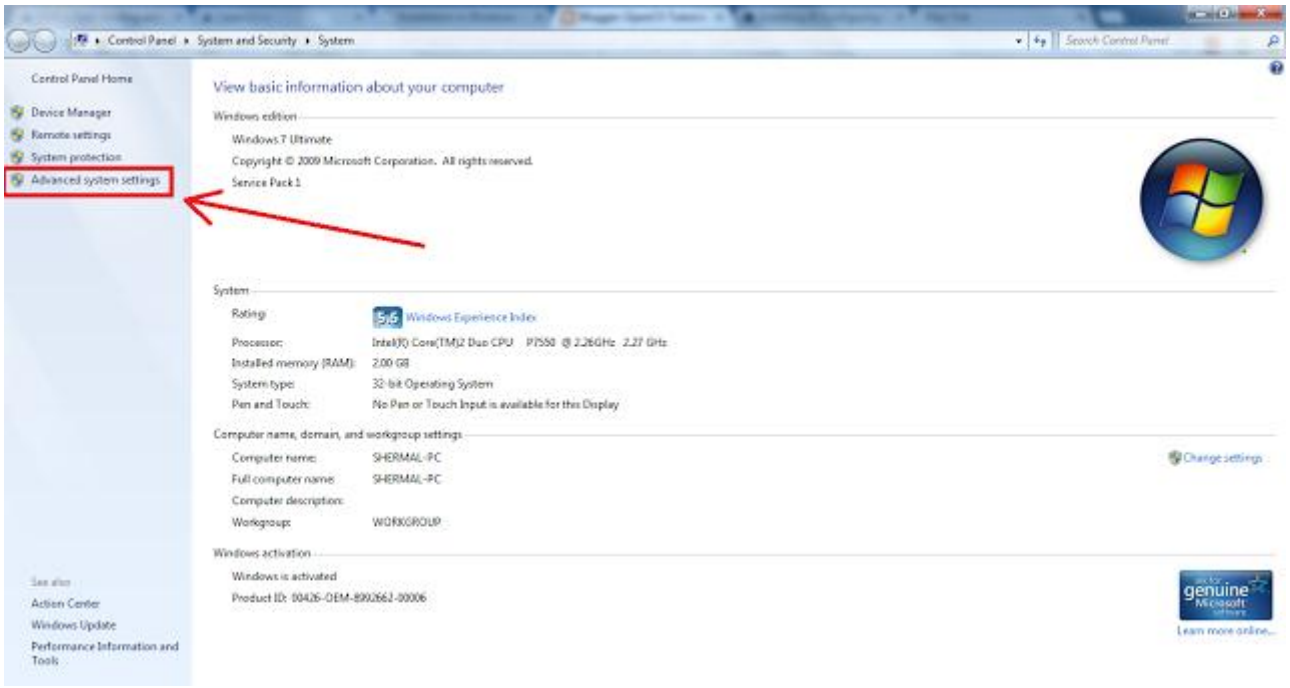
Aquí el tipo de sistema es el sistema operativo de **32 bits**. Así que la arquitectura del

sistema es **x86**. Si el tipo de sistema es el sistema operativo de **64 bits**, la arquitectura del sistema es **x64**.

Tanto Visual Studio 2012 como OpenCV2.4.5 admiten vc11. Por lo tanto, en este tutorial, voy a utilizar vc11 como el tipo de compilador. Puede encontrar los tipos de compilador compatibles de OpenCV, si va a C: \ opencv \ build \ x86 o C: \ opencv \ build \ x64 de acuerdo con la arquitectura de su sistema.

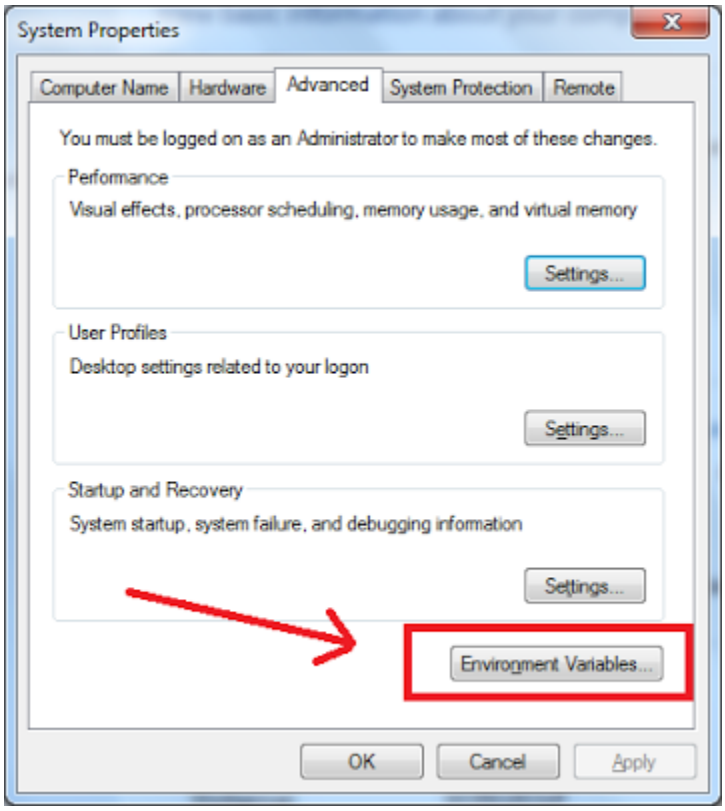
Esto se puede hacer en la línea de comandos o mediante GUI.

Hacer clic derecho en ' **Mi PC** ' y luego en ' **Propiedades** ' en el menú desplegable. Se verá una ventana como esta.

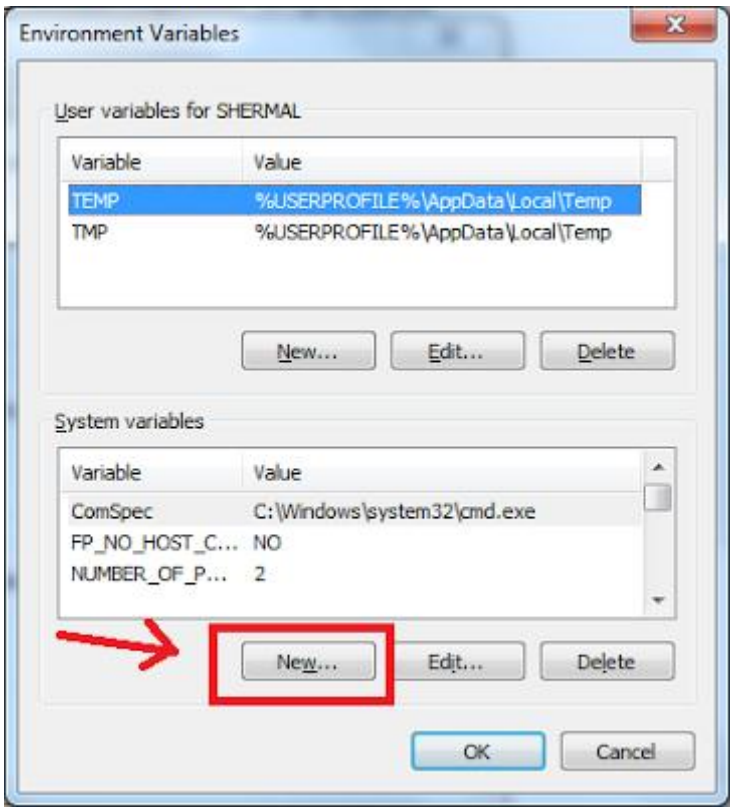


Haga clic en " **Avanzar configuración del sistema"** en la ventana anterior

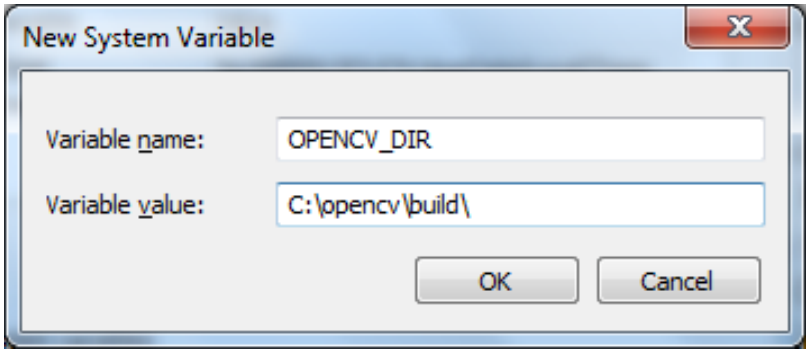
A continuación, hacer clic en **"Variables de entorno"**



A continuación, haga clic en **el** botón **"Nuevo"** en la parte inferior de la ventana



Tipo **OPENCV_DIR** contra el **Nombre de la variable**. Y escribir la ubicación **C: \ opencv \ build ** contra el **valor Variable.**

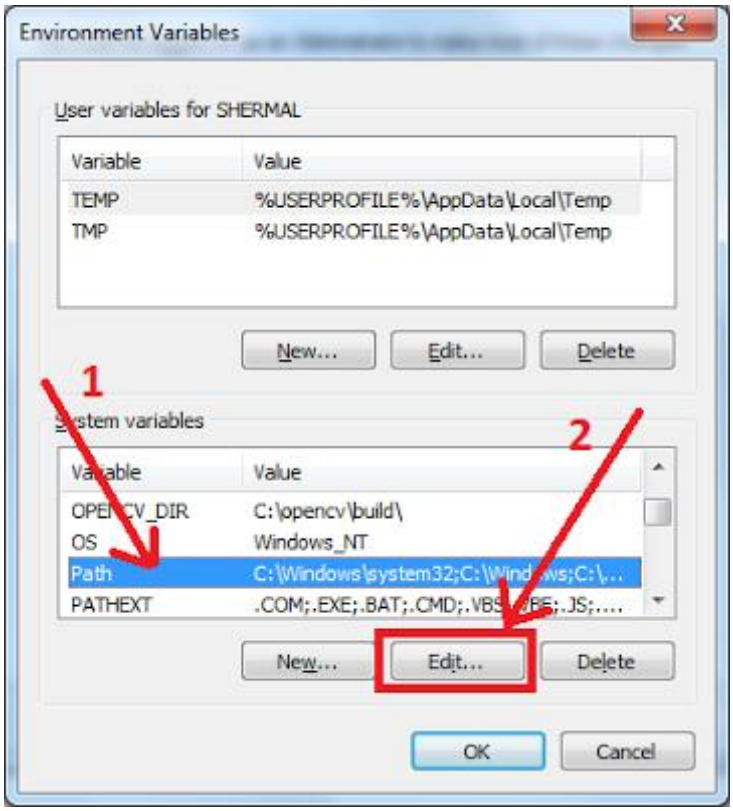


A continuación, pulsar **OK**

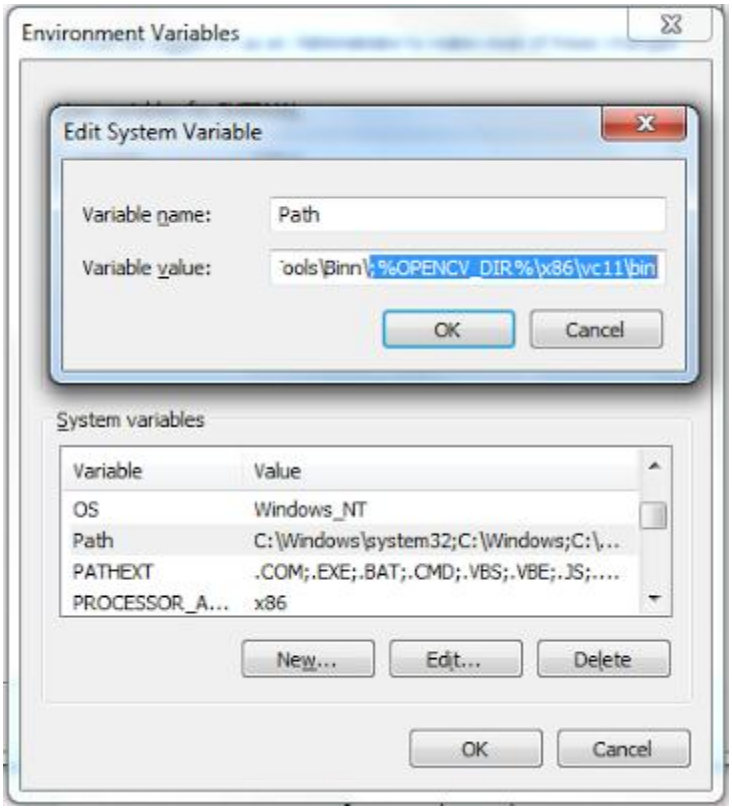
Ahora ha agregado una nueva variable de entorno al sistema.

Ahora se edita una variable del sistema.

Haga clic en **'PATH'** dentro de la lista de variables del sistema y luego hacer clic en **el** botón **'Editar'** en la parte inferior de la ventana



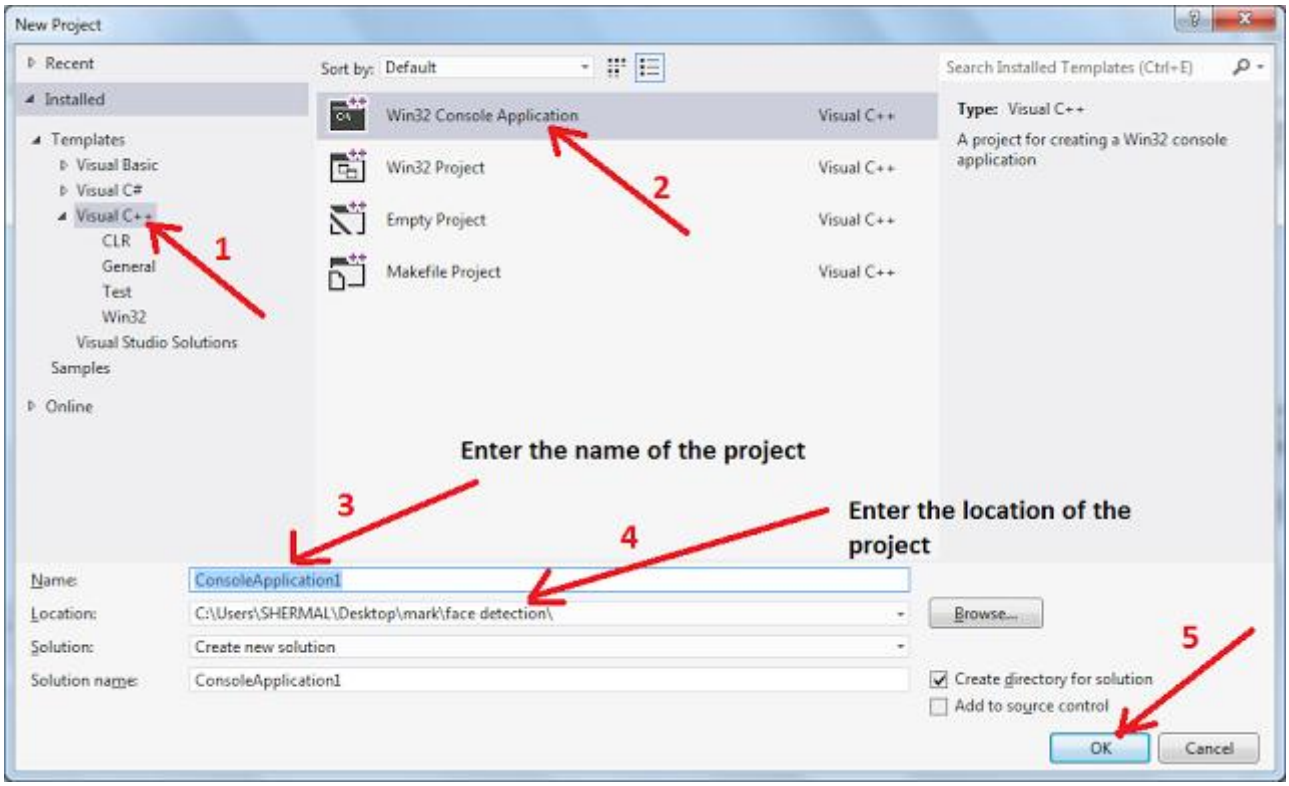
A continuación, agregar un; al final de la línea agregar lo siguiente % OPENCV_DIR% \ x86 \ vc11 \ bin, después del punto y coma. X86 es la arquitectura del sistema y vc11 es el tipo de compilador.



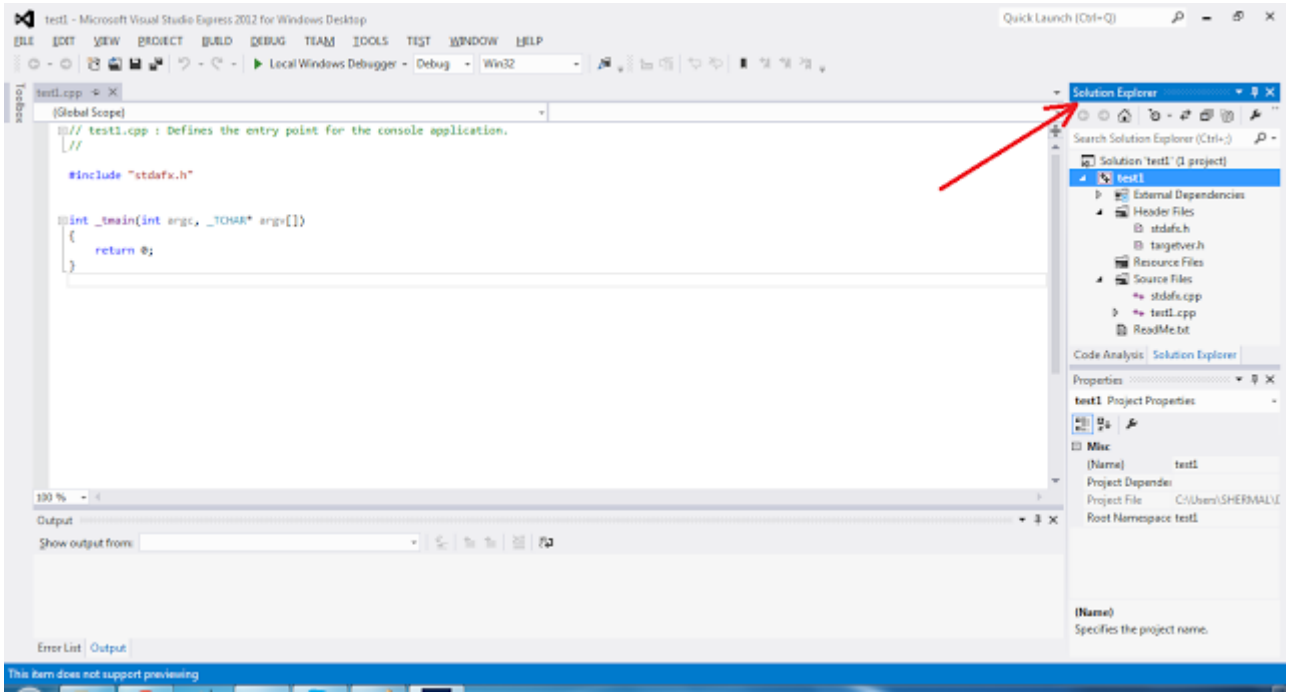
Ahora hay que configurar Visual Studio.

Configurar Visual Studio Express

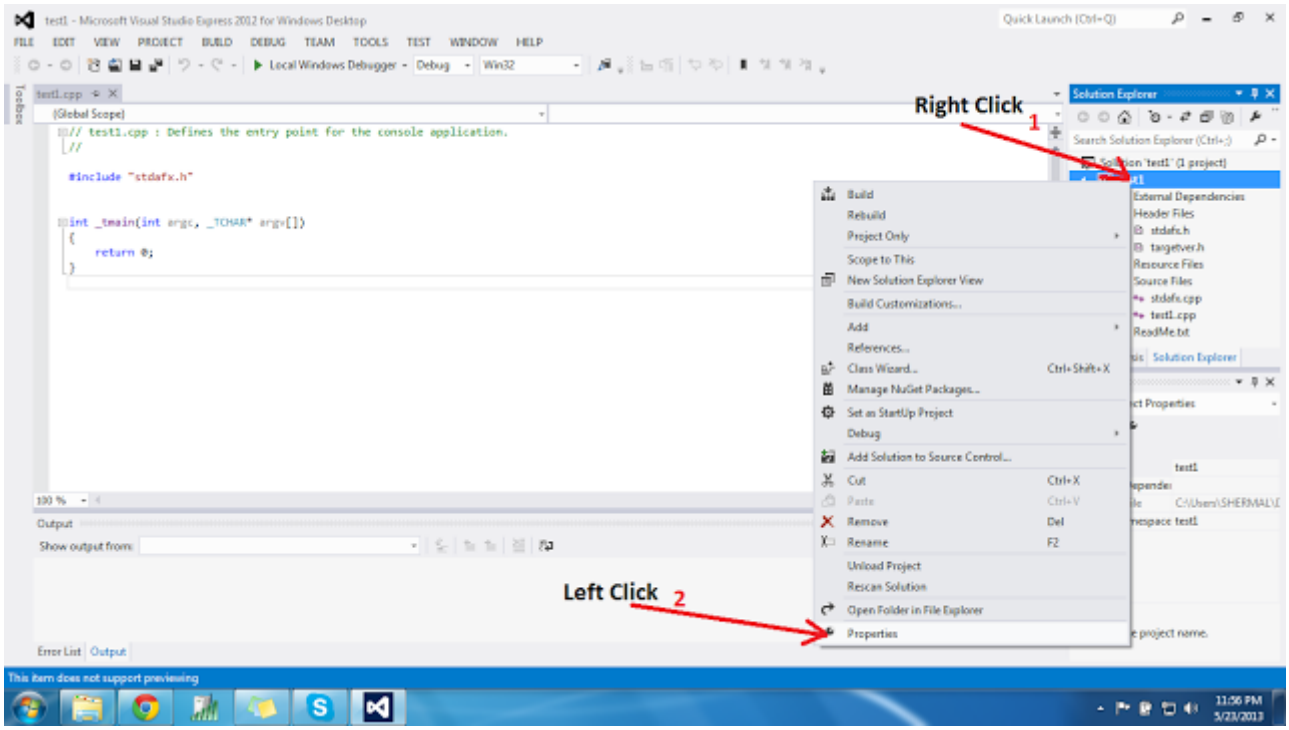
- Iniciar Microsoft Visual Studio
- Ir a Archivo> Nuevo proyecto ...
- Hacer todo lo que se muestra en la imagen de abajo y hacer clic en Aceptar y, a continuación, hacer clic en Finalizar.



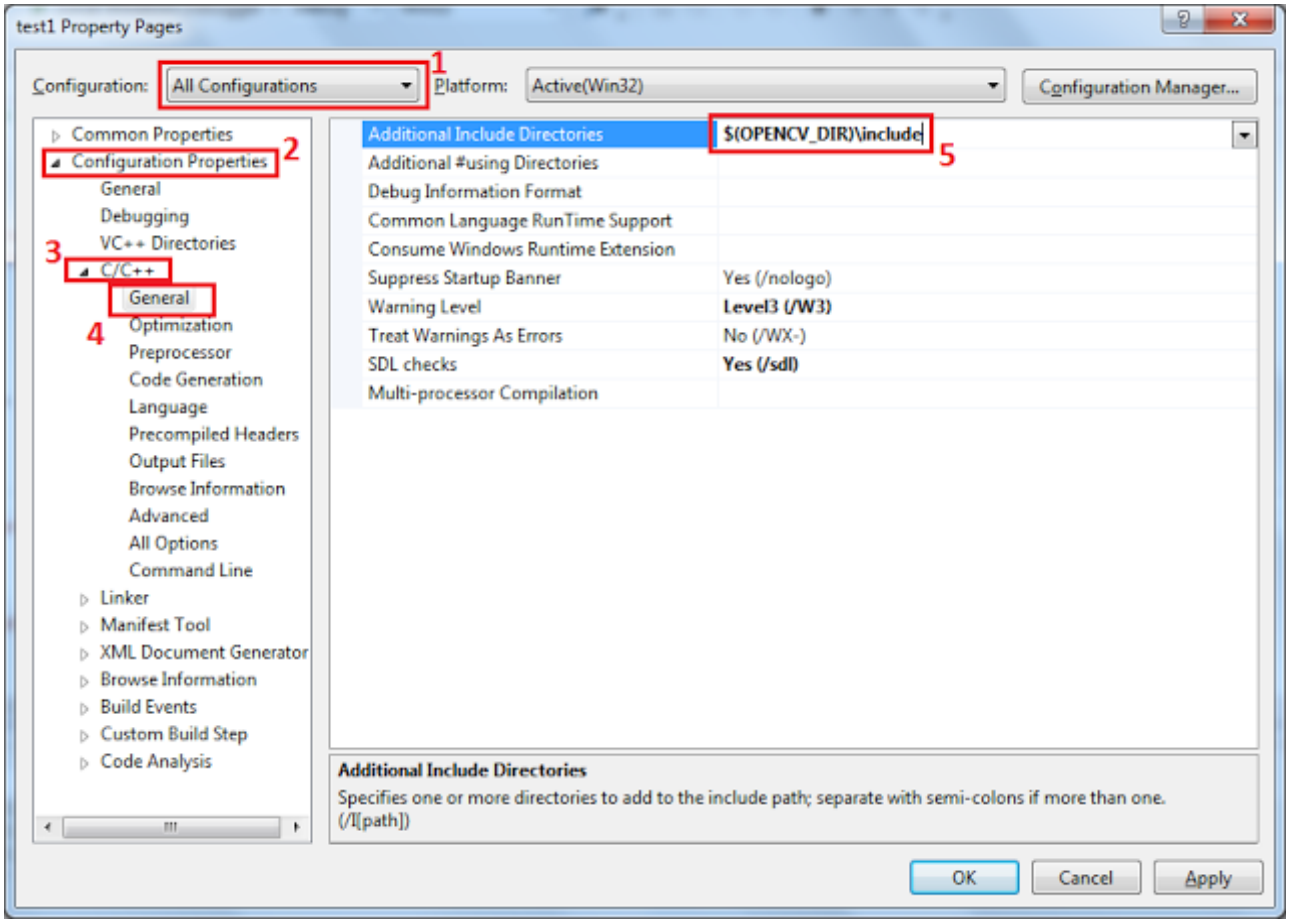
- Presionar la **tecla** ' **Ctrl** ' y luego ' **W** ' mientras mantiene presionada la tecla 'Ctrl' en su teclado. Soltar ambas teclas. A continuación, pulsar '**S**' en el teclado. A continuación, se verá el panel " **Explorador de soluciones** " en el lado derecho o izquierdo de la ventana de Visual Studio.



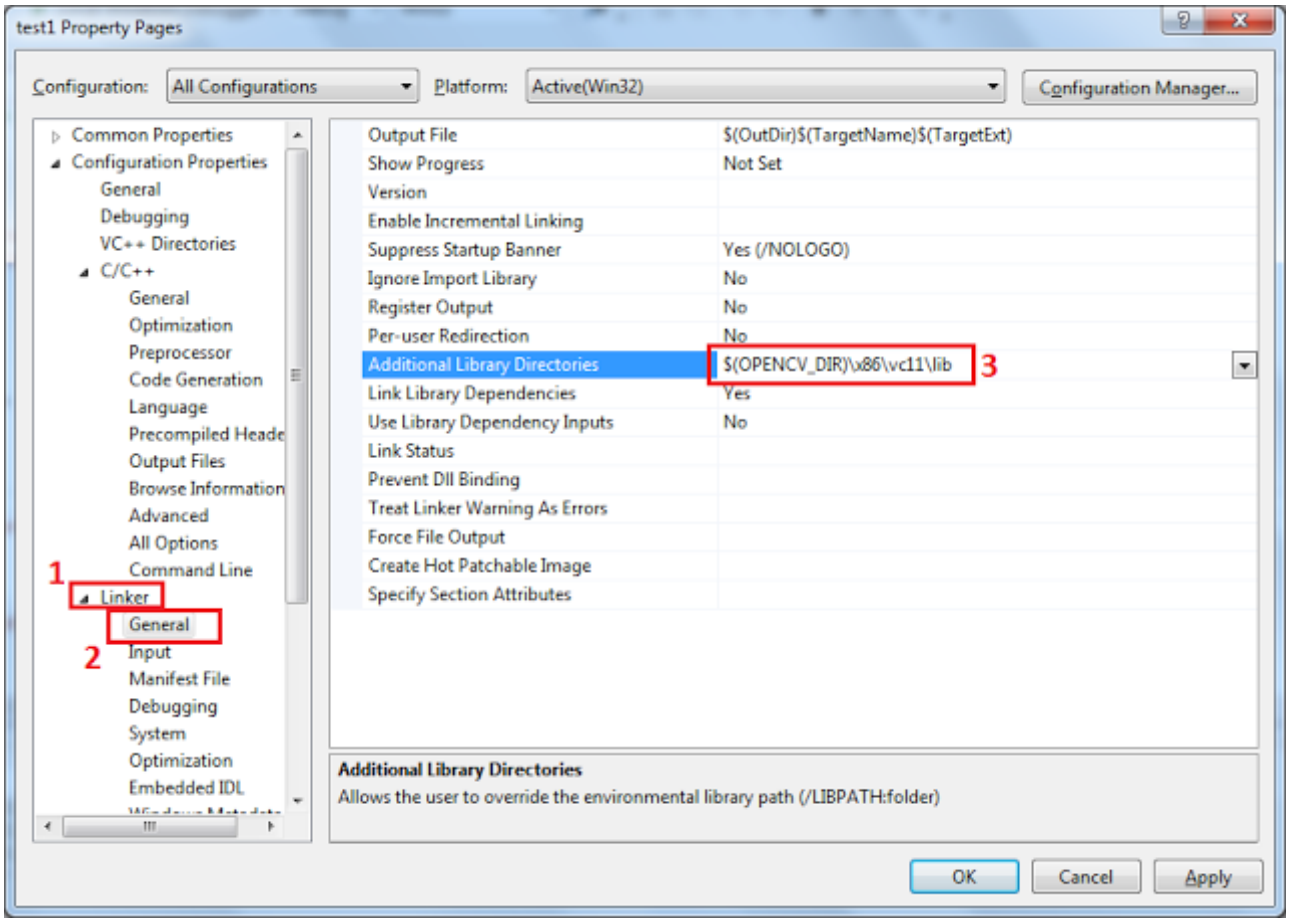
- A continuación, hacer clic con el botón secundario en el nombre del proyecto (ha introducido este nombre en un paso anterior) y, a continuación, hacer clic en " **Propiedades** "



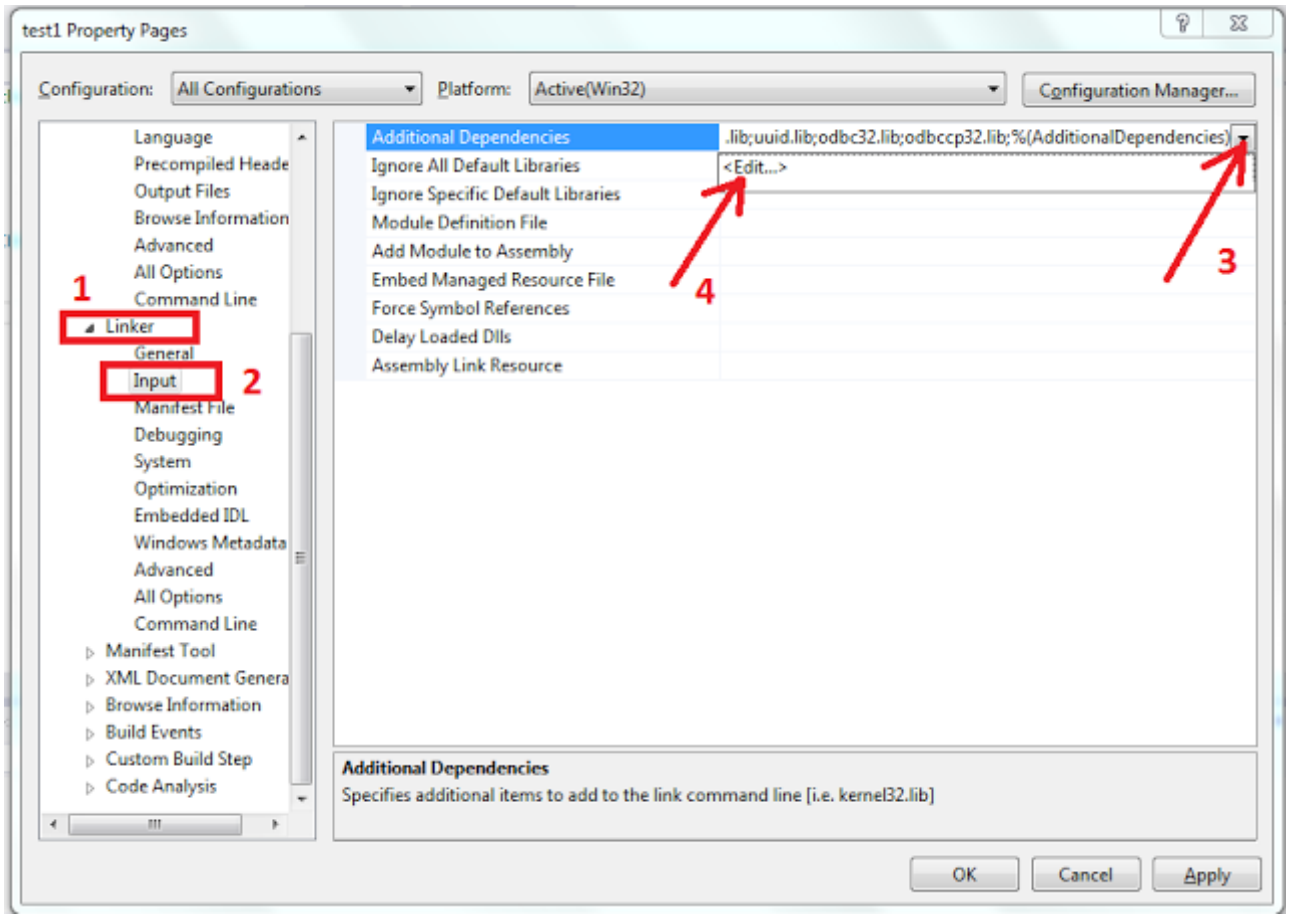
- Seleccionar 'Todas las configuraciones' como se indica en la 1ª casilla. Hacer clic en los 3 lugares indicados por 3 cuadros siguientes en el orden dado. A continuación, copiar y pegar $ (OPENCV_DIR) \ include en la opción de 'Incluir Directorios Adicionales'



- Hacer clic en los primeros 2 lugares como se indica por los primeros 2 cuadros en el orden dado. Luego, copiar y pegar $(OPENCV_DIR) \ x86 \ vc11 \ lib en la opción "Directorios de biblioteca adicionales"



Hacer clic como se muestra a continuación



Después de hacer clic en <Editar ...> (4ª flecha en la imagen anterior), aparecerá un cuadro de diálogo y se tendrá que copiar y pegar los siguientes nombres de archivo de biblioteca. Si se está utilizando una versión diferente de OpenCV que no sea 2.4.5, comprobar la ubicación 'C: \ opencv \ build \ x86 \ vc11 \ lib' para los nombres de archivo respectivos.

Opencv_calib3d245d.lib

Opencv_contrib245d.lib

Opencv_core245d.lib

Opencv_features2d245d.lib

Opencv_flann245d.lib

Opencv_gpu245d.lib

Opencv_haartraining_engined.lib

Opencv_highgui245d.lib

Opencv_imgproc245d.lib

Opencv_legacy245d.lib

Opencv_ml245d.lib

Opencv_nonfree245d.lib

Opencv_objdetect245d.lib

Opencv_photo245d.lib

Opencv_stitching245d.lib

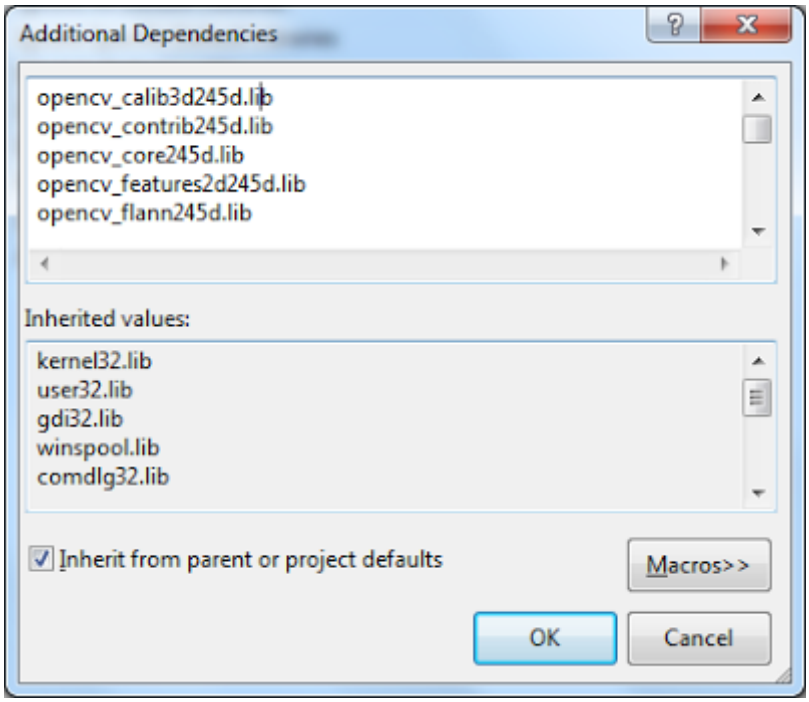
Opencv_superres245d.lib

Opencv_ts245d.lib

Opencv_video245d.lib

Opencv_videostab245d.lib

Estos son algunos de los nombres de archivo, que se puede encontrar en la ubicación 'C: \ opencv \ build \ x86 \ vc11' .



Ahora hacer clic en Aceptar

PROGRAMA PRINCIPAL

```
#include <opencv2\opencv.hpp>
#include "OCR_ZON_PROF_SIM_COS.h"
using namespace cv;
using namespace std;

int main()
{
	string strMatricula = string[8];
	Mat img_gris, img_out, src,frame;
	Leer_Archivo();
	VideoCapture cap;
	cap.open(1);
	cap.set(CV_CAP_PROP_FRAME_WIDTH,640);
	cap.set(CV_CAP_PROP_FRAME_HEIGHT,480);
	if(!cap.isOpened())
		return -1;
	namedWindow("webcam");

	while(1)
	{
		cap.read(frame);
		//
**************************************************************
		// PREPROCESAMIENTO
**************************************************************
		cvtColor(frame, img_gris,CV_BGR2GRAY);

		threshold(img_gris, img_out, 100.0, 255,CV_THRESH_BINARY_INV);
getStructuringElement(MORPH_RECT,Size(8,1)),Point(-1,-1),1);
		namedWindow("img_out",1);
```

```
imshow("img_out",img_out);

// ENCONTRAMOS CONTORNOS DE CARACTERES
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
Mat img_Placa;
img_out.copyTo(img_Placa );
findContours(img_Placa,
contours,hierarchy,CV_RETR_EXTERNAL,CV_CHAIN_APPROX_SIMPLE);
CvRect rectP[10];

putText(frame,"Caract: " + contours.size(), Point(400,470),
FONT_ITALIC,0.9,Scalar(255,0,0),2);

for( int i = 0; i< contours.size(); i++ )
{
    double ar=contourArea(contours,false);
    if ( ar > 30 && ar < 50 )
    {
        rectP[i] = boundingRect(contours);
        rectangle(frame,rectP[i],Scalar(0,0,255));
        Mat c1P(25, 15, CV_8UC1 );
        Size size(15,25);
        Mat c1 = img_out(rectP[i]);
        resize(c1, c1P,size,0,0,CV_INTER_CUBIC);
        threshold(c1P, c1P, 100.0, 255, CV_THRESH_BINARY);

        strMatricula = Reconocer(c1P, 0);
        putText(frame, strMatricula[0], Point(rectP[i].x, rectP[i].y-
5), FONT_ITALIC,0.7,Scalar(0,255,0),2);
        //
    }
}
```

```
            imshow("webcam",frame);
            if(waitKey(10) == 27) break;           // ESC
        }
        return 0;
}
```

FUNCIÓN DE RECONOCIMIENTO

```
int Leer_Archivo()
{
        int xx = 0;
        for(int dir = 0; dir < 10; dir++)
        {
            for(int arch = 0; arch < 5; arch++)
            {
                char nombre_archivo_leer[100];

                sprintf(nombre_archivo_leer,"E:/
L_Caracteres/%d/cap_%d.jpg",dir,arch);
                Mat img1 = imread(nombre_archivo_leer,0);

                if (img1.empty())
                    return -1;

                threshold(img1, img1, 100.0, 255,CV_THRESH_BINARY);

                Mat vectProf1=Mat::zeros(1,15,CV_32F);
                Mat vectProf2=Mat::zeros(1,80,CV_32F);

                int cont1 = 0, cont2 = 0, cont3 = 0;

                for(int f = 0; f < 5; f++ )
                {
                    for(int c = 0; c < 5; c++)
```

```
                    if(img1.at<uchar>(f,c) > 0) cont1++;

            for(int c = 5; c < 10; c++)
                    if(img1.at<uchar>(f,c) > 0) cont2++;

            for(int c = 10; c < 15; c++)
                    if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(0) = cont1;
vectProf1.at<float>(1) = cont2;
vectProf1.at<float>(2) = cont3;

cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 5; f < 10; f++ )
{
            for(int c = 0; c < 5; c++)
                    if(img1.at<uchar>(f,c) > 0) cont1++;

            for(int c = 5; c < 10; c++)
                    if(img1.at<uchar>(f,c) > 0) cont2++;

            for(int c = 10; c < 15; c++)
                    if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(3) = cont1;
vectProf1.at<float>(4) = cont2;
vectProf1.at<float>(5) = cont3;

cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 10; f < 15; f++ )
{
```

```
        for(int c = 0; c < 5; c++)
                if(img1.at<uchar>(f,c) > 0) cont1++;

        for(int c = 5; c < 10; c++)
                if(img1.at<uchar>(f,c) > 0) cont2++;

        for(int c = 10; c < 15; c++)
                if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(6) = cont1;
vectProf1.at<float>(7) = cont2;
vectProf1.at<float>(8) = cont3;

cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 15; f < 20; f++ )
{
        for(int c = 0; c < 5; c++)
                if(img1.at<uchar>(f,c) > 0) cont1++;

        for(int c = 5; c < 10; c++)
                if(img1.at<uchar>(f,c) > 0) cont2++;

        for(int c = 10; c < 15; c++)
                if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(9) = cont1;
vectProf1.at<float>(10) = cont2;
vectProf1.at<float>(11) = cont3;

cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 20; f < 25; f++ )
```

```
{
        for(int c = 0; c < 5; c++)
                if(img1.at<uchar>(f,c) > 0) cont1++;

        for(int c = 5; c < 10; c++)
                if(img1.at<uchar>(f,c) > 0) cont2++;

        for(int c = 10; c < 15; c++)
                if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(12) = cont1;
vectProf1.at<float>(13) = cont2;
vectProf1.at<float>(14) = cont3;

for(int k = 0; k < 15; k++)     // NORMALIZANDO
{
        P_map[xx][k] = vectProf1.at<float>(k);
}
for(int f = 0; f < 25; f++ )
{
        for(int c = 0; c < 15; c++)
        {
                if(img1.at<uchar>(f,c)  ==  0)      // Si pixel es
Background
                        vectProf2.at<float>(f) += 1;
                else
                        break;
        }
}
for(int f = 0; f < 25; f++ )
{
        for(int c = 14; c >= 0; c--)
        {
```

```
                    if(img1.at<uchar>(f,c) == 0)      // Si pixel es
Background
                        vectProf2.at<float>(f+25) += 1;
                    else
                        break;
                }
            }

            for(int c = 0; c < 15; c++ )
            {
                for(int f = 0; f < 25; f++)
                {
                    if(img1.at<uchar>(f,c) == 0)      // Si pixel es
Background
                        vectProf2.at<float>(c+50) += 1;
                    else
                        break;
                }
            }
            for(int c = 0; c < 15; c++ )
            {
                for(int f = 14; f >= 0; f--)
                {
                    if(img1.at<uchar>(f,c) == 0)      // Si pixel es
Background
                        vectProf2.at<float>(c+65) += 1;
                    else
                        break;
                }
            }

            for(int k = 0; k < 80; k++)
            {
```

```
                    P_map[xx][k+15] = vectProf2.at<float>(k);
                }

                xx++;

            }
        }
}

char Reconocer(Mat img1, int num)
{
char Caracteres[10]={'0','1','2','3','4','5','6','7','8','9','?'};

    Mat vectProf1=Mat::zeros(1,15,CV_32F);
    Mat vectProf2=Mat::zeros(1,80,CV_32F);

    int cont1 = 0, cont2 = 0, cont3 = 0;
    int R_map[95]={0};

    for(int f = 0; f < 5; f++ )
    {
        for(int c = 0; c < 5; c++)
            if(img1.at<uchar>(f,c) > 0) cont1++;

        for(int c = 5; c < 10; c++)
            if(img1.at<uchar>(f,c) > 0) cont2++;

        for(int c = 10; c < 15; c++)
            if(img1.at<uchar>(f,c) > 0) cont3++;
    }
    vectProf1.at<float>(0) = cont1;
    vectProf1.at<float>(1) = cont2;
    vectProf1.at<float>(2) = cont3;
```

```
cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 5; f < 10; f++ )
{
	for(int c = 0; c < 5; c++)
		if(img1.at<uchar>(f,c) > 0) cont1++;

	for(int c = 5; c < 10; c++)
		if(img1.at<uchar>(f,c) > 0) cont2++;

	for(int c = 10; c < 15; c++)
		if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(3) = cont1;
vectProf1.at<float>(4) = cont2;
vectProf1.at<float>(5) = cont3;

cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 10; f < 15; f++ )
{
	for(int c = 0; c < 5; c++)
		if(img1.at<uchar>(f,c) > 0) cont1++;

	for(int c = 5; c < 10; c++)
		if(img1.at<uchar>(f,c) > 0) cont2++;

	for(int c = 10; c < 15; c++)
		if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(6) = cont1;
vectProf1.at<float>(7) = cont2;
```

```
vectProf1.at<float>(8) = cont3;

cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 15; f < 20; f++ )
{
        for(int c = 0; c < 5; c++)
                if(img1.at<uchar>(f,c) > 0) cont1++;

        for(int c = 5; c < 10; c++)
                if(img1.at<uchar>(f,c) > 0) cont2++;

        for(int c = 10; c < 15; c++)
                if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(9) = cont1;
vectProf1.at<float>(10) = cont2;
vectProf1.at<float>(11) = cont3;

cont1 = 0; cont2 = 0; cont3 = 0;

for(int f = 20; f < 25; f++ )
{
        for(int c = 0; c < 5; c++)
                if(img1.at<uchar>(f,c) > 0) cont1++;

        for(int c = 5; c < 10; c++)
                if(img1.at<uchar>(f,c) > 0) cont2++;

        for(int c = 10; c < 15; c++)
                if(img1.at<uchar>(f,c) > 0) cont3++;
}
vectProf1.at<float>(12) = cont1;
```

```
vectProf1.at<float>(13) = cont2;
vectProf1.at<float>(14) = cont3;

for(int k = 0; k < 15; k++)
{
	R_map[k] = vectProf1.at<float>(k);
}
for(int f = 0; f < 25; f++ )
{
	for(int c = 0; c < 15; c++)
	{
		if(img1.at<uchar>(f,c) == 0)   // Si pixel es Background
			vectProf2.at<float>(f) += 1;
		else
			break;
	}
}
for(int f = 0; f < 25; f++ )
{
	for(int c = 14; c >= 0; c--)
	{
		if(img1.at<uchar>(f,c) == 0)   // Si pixel es Background
			vectProf2.at<float>(f+25) += 1;
		else
			break;
	}
}

for(int c = 0; c < 15; c++ )
{
	for(int f = 0; f < 25; f++)
	{
		if(img1.at<uchar>(f,c) == 0)   // Si pixel es Background
```

```
                    vectProf2.at<float>(c+50) += 1;
                else
                    break;
            }
        }
        for(int c = 0; c < 15; c++ )
        {
            for(int f = 14; f >= 0; f--)
            {
                if(img1.at<uchar>(f,c) == 0)
                    vectProf2.at<float>(c+65) += 1;
                else
                    break;
            }
        }

        for(int k = 0; k < 80; k++)
        {
            R_map[k+15] = vectProf2.at<float>(k);
        }

        double s1 = 0, s2 = 0, s3 = 0;

        double maxValor1=0, maxValor2=0, maxValor3=0, UMBRAL = 0.8, maxval =
0;

        int caracterX1 = 0, caracterX2 = 0, caracterX3 = 0, caracterX = 0;

        for(int xx = ini;xx < fin; xx++)
        {
            s1 = 0; s2 = 0; s3 = 0;
            for(int k = 0; k < 15; k++)
            {
```

```
                    s1 = s1 + P_map[xx][k] * R_map[k];
                    s2 = s2 + P_map[xx][k] * P_map[xx][k];
                    s3 = s3 + R_map[k] * R_map[k];
                }
                s2 = sqrt(s2);
                s3 = sqrt(s3);
                maxval = s1 / (s2 * s3);
                if (maxval > maxValor1)
                {
                    maxValor1 = maxval;
                    caracterX1 = xx;
                }
            }

            caracterX1 = (int) (caracterX1 / 5.0);

            if (maxValor1 > UMBRAL)
                return Caracteres[caracterX1];
    }
```